Proceedings of the 14th IFAC Symposium on
Information Control Problems in Manufacturing
Bucharest, Romania, May 23-25, 2012

A Model-Based Approach for Planning Work Breakdown Structures of Complex Systems Projects

Amira Sharon*, Dov Dori**

* Technion, Israel Institute of Technology, Haifa 32000, Israel (Tel: +972503662466; e-mail: amirash@technion.ac.il).
** Technion, Israel Institute of Technology, Haifa 32000, Israel, and Massachusetts Institute of Technology, Cambridge, MA (e-mail: dori@mit.edu)}

Abstract: Planning the development efforts within large-scale projects is a highly complex mission due to uncertainties regarding different aspects of the product to be delivered. Current planning practices employ a host of methods for project planning, with Work Breakdown Structure (WBS) being a prominent one of them. Using WBS, the deliverable—the end product to be delivered, its components, and associated enabling products—are often induced implicitly. Using a running example of an unmanned aerial vehicle, we review the WBS method and discuss problems stemming from the lack of explicit and direct representation of the product facet in the project plan. Based on this observation, we apply the Project-Product Lifecycle Management (PPLM) approach as an exclusive source of the various enhanced project management tools, which are views of the model. WBS, the focus of this paper, is one of these views. Like the other views, our WBS version is augmented with product-related information gleaned from the common underlying model.

Keywords: Complex Systems, Project Planning, Work Breakdown Structure, Model-Based Planning.

1. INTRODUCTION

The project and the product it delivers are inseparable. What needs to be developed, tested, and delivered is determined by the product requirements, its architecture, and design. Each component should and can be developed and tested is stated in the project plan. A framework and process for integrating perspectives of a developed complex system with its corresponding project processes is clearly missing (Cook et al., 2003). Furthermore, empirical investigations have shown that the relationships and interactions between the architecture of products, their development projects, and the organizational teams involved, should be aligned in order for a company to become successful (Eppinger and Salminen, 2001). However, since the development of a complex system involves design processes, which differ from workflow processes (Fischer, 2005), the planned activities are usually unique to each specific project. Moreover, there is no single "correct" way for the breaking down the project activities. In large-scale projects, the project and the product are decomposed hierarchically. The output of each domain – the project domain and the product domain – evolves gradually and hierarchically from an abstract concept into detailed information. The hierarchical decomposition in one domain cannot be performed independently of the other domains, i.e., this decomposition requires zigzagging in order to map adjacent domains to each other. Sim and Duffy (2003) classified engineering design activities from published literature into three categories – design definition activities, design evaluation activities, and design management activities. The combined project-product planning fits into the design definition activities category. The planning process is aimed at managing the complexity of the evolving design while increasingly defining it, until it has all the required details. The planning process involves continuous processing of information within and between the project domain and the product domain. As this decomposition takes place, links between each subsystem (or each system in a system-of-systems scenario) and the corresponding project must be maintained, resulting in an intricate network of relationships between the product and the project. This network is difficult to maintain without a common underlying model, and is the heart of the problem that the Project-Product Lifecycle Management (PPLM) approach (Sharon et al., 2011) addresses. The integrated PPLM approach and the model used in this paper induces traceability, explicating critical relationships between the product and the project. As we show in the sequel, the ability to simultaneously express the required information from the project and the product domains within a single integrated model-based framework can potentially lead to a more reliable project plan, which is less prone to the need for repeated changes, rework, and corrective actions. The increased robustness of the resulting project plan is attributed to the need to consult the product model integrated into the PPLM model while making decisions about the project's "technological order" (Levy et al., 1963). The planning process carried out following this approach clarifies the intricate relationships between project and product entities. This model-based approach enables the simultaneous expression of the function, structure and behaviour of both the project and the product via the same

978-3-902661-98-2/12/$20.00 © 2012 IFAC

10.3182/20120523-3-RO-2023.00255

ontological and methodological foundations, maintaining full traceability between project and product data.

We focus on the Work Breakdown Structure (WBS), a widely used planning method (Mil-Std-881, 1968). A WBS is a hierarchical tree structure decomposition of a project into smaller components down to the level of work elements—the leaves of the hierarchy. WBS is supposed to reflect the total scope of work involved in the project. Capturing the total work and efforts required for the project is most important, as WBS is the source for project cost estimations, schedule planning, and risk mitigation. The Guide to the Project Management Body of Knowledge (PMBOK®, 2008) promotes a deliverables oriented decomposition of the work to be executed in order to accomplish the project objectives and produce the anticipated deliverables.

In order to produce a deliverables-oriented WBS, the WBS has to be based on planned outcomes throughout the project, rather than on planned actions. Using this approach produces a project plan in which the product is written all over, albeit not explicitly. Following the introduction of the Project-Product Lifecycle Management (PPLM) approach, we propose a model-based counterpart of WBS and demonstrate its advantages with respect to the classical WBS.

2. THE PROJECT-PRODUCT LIFECYCLE MANAGEMENT FRAMEWORK

While Product Lifecycle Management (PLM) is a concept that has been around for two decades, our Project-Product Lifecycle Management (PPLM) approach (Sharon et al., 2011), calls for constructing a comprehensive product-project model. This model integrates the project domain with the product domain via a shared ontology, utilizing OPM – Object Process Methodology (Dori, 2002). The potential users of our PPLM framework come from a wide range of disciplines and user profiles, including enterprise, project, and systems engineering managers. The notation, while being formal, must therefore also be simple and intuitive, so all the potential users and stakeholders can relate to the model as it evolves.

A careful comparison of four candidate modeling languages - UML, xUML, SysML, and OPM – led to selection of OPM as the underlying conceptual modeling language and paradigm for PPLM. OPM (Dori, 2001) is a holistic, integrated approach to the design and development of systems in general and complex dynamic systems in particular. OPM is a formal yet intuitive paradigm for systems architecting, engineering, development, lifecycle support, and evolution. It has been used for modeling complex systems, both natural and artificial, where artificial ones might comprise humans, physical objects, hardware, software, regulations, and information. As its name suggests, the two basic building blocks in OPM are (stateful) objects—things that exist (at some state), and processes—things that transform objects by creating or destroying them, or by changing their state. Objects and processes are of equal importance, as they complement each other in the single-model specification of the system. Links, which are the OPM elements that connect entities, are of two types: structural and procedural. The generic definitions of OPM elements makes it suitable for modeling complex systems that comprise technology and humans—this is the type of systems that aim to deliver complex products via executing large-scale projects.

OPM notation supports conceptual modeling of systems using a single type of diagram to describe the functional, structural and behavioural aspects of a system. An OPM model consists of a set of hierarchically organized Object-Process Diagrams (OPDs) that alleviate systems' complexity. Each OPD is obtained by in-zooming or unfolding of a thing (object or process) in its ancestor OPD. OPCAT (Dori et al., 2003), an OPM-based conceptual modeling software environment, features an accessible API, a basic animated class-level execution module, and integration with files of various formats, e.g., XML and CSV, reducing the development effort.

OPM is currently in the process of becoming an ISO standard for enterprise standards. When completed, this endorsement will enable accelerated dissemination of OPM as a basis for enterprise standards in general and for PPLM in particular.

Using OPM, the combined project-product model ultimately contains the activities (processes) and the deliverables (objects). Activities and tasks are OPM processes at various detail levels required for completing the product. Deliverables are product components, resources and informatical objects, such as documents and approvals. Having all this information embedded consistently in the same PPLM model eventually yields all the specific structural relations (among objects) and procedural relations (between objects and processes). Understanding the product's structure hierarchy hand-in-hand with the project activities and progress provides the basis for the technological process order and timing. With this understanding, the project planner can model the rationale for ordering the project activities—the model processes—by identifying the flow of objects into and out of each process.

3. THE MODEL-BASED PROJECT-PRODUCT PLAN

3.1 The Unmanned Aerial Vehicle Project Plan

As a case in point, we consider a project of developing a simplified Unmanned Aerial Vehicle (UAV) by an imaginary New Millennium Aerospace (NMA) Inc., a government-contracted leading UAVs manufacturer. A rough specification of the UAV concept (de Weck and Lyneis, 2008) was used as the basis for the UAV development, which focuses on the vehicle, while the payload is provided by the government as modified government furnished equipment (GFE), and the engine is supplied by an established commercial company (ECC) under a subcontract. The UAV specification, given in Fig. 1, contains a description of 23

tasks necessary to develop the UAV, including the dependencies between tasks and their durations. Each task ("job") description is underlined the first time it is mentioned, and the task ID and its normal duration in weeks are given in parentheses.

After the project start (a, 0) you first have to complete the overall requirements definition (b, 10) step. Once this is accomplished you can carry out the following jobs in parallel: negotiate the engine specification (c, 5) with ECC, define your payload specification (d, 5), determine the vehicle layout (e, 8) and write the software specification (g, 12). You can initiate (GFE) avionics design (f, 15) after (b, 10), however the tasks (c, 5) and (d, 5) must also have been completed before the GFE design can be started, so that the avionics will be able to control both the engine and payload in a synchronized fashion.
Once the engine specs (c, 5) have been defined, the supplier (ECC) informs you that it will take 30 weeks for engine development (i, 30) based on experience with a previous variant. Once engine development is complete, delivery and checkout (n, 2) can take place at NMA's facilities. After (d, 5) is done, payload development (j, 15) can take place in parallel with engine development. Once the payload is developed (j, 15) and the engine delivered (n, 2), both the engine and payload are integrated (electrically) in the power system integration (o, 10) step. Fuselage design (k, 17) and empennage/wing design (l, 15) begin in parallel after the vehicle layout (e, 8) has been established. Internal fittings (m, 8) can be designed after these two jobs are completed. Also, structural airframe prototyping (r, 8) consists of building a physical frame for the vehicle after jobs (k, 17) and (l, 15) are completed. Once avionics design (f, 15) has been completed, this leads to avionics delivery and checkout (p, 12) and subsequent avionics/software integration (q, 5). Obviously, in order for this last step to take place, software development (h, 25) which depends both on (g, 12) and (f, 15) must have also been completed. The project is continued by performing vehicle integration (s, 10), which requires prior completion of power system integration (o, 10), airframe prototyping (r, 8) and avionics/software integration (q, 5). After vehicle integration (s, 10) and internal fitting design (m, 8) have been achieved, final vehicle assembly (t, 5) can begin. After final assembly, the completed vehicle is subjected to laboratory testing (u, 5), followed by an outdoor flight test campaign (v, 10), leading to completion of the prototype development project, finish (w, 0).

Notes:
- task descriptions are underlined
- (n, 25) means that the task is tagged as "n" and is expected to take 25 work weeks. For example, engine integration (x, 17) means that there is a task called "engine integration," identified by the symbol "x," whose nominal duration is 17 weeks.
- task descriptions are hypothetical, but in a notionally meaningful sequence
- task durations are hypothetical (on the short side)

Fig. 1. Basic specification of the UAV development concept

3.1 The OPM Notation for Model-Based Project Planning

The OPM-based project-product plan was modelled using OPCAT (Dori et al., 2003). The model is based strictly on the text in Fig. 1, and deliverables were added based on this text. The activities identified in the specification are OPM processes in the model, while all the inputs and outcomes of these activities are OPM objects. The **Agents Set** (the human enablers) and the **Instruments Set** (the non-human enables) were also modelled based on the text. The duration of each task is the duration of the corresponding task (leaf-level process), assigned in OPCAT using the minimum activation time attribute of the process. The activation time units can be set to milliseconds, seconds, minutes, hours, days, months, or years. Following Fig. 1, the activation time for each process was defined in weeks. For the sake of simplicity, Budget is not included in the current model as a consumed input, but it can be readily added.

The model follows guidelines of the methodology part of OPM. It starts with a top-level process at the System Diagram (SD, top-level OPD) of the model, representing the system's function, which in our case is the execution of the entire project. The model than contains decomposition of the system hierarchy using OPM refinement mechanisms, primarily in-zooming, following the top-to-bottom ordering of a sequence of sub-processes within an in-zoomed process.

The entire project is modelled through concurrent hierarchical decomposition of processes and objects involved in these processes as resources (inputs or enablers) or deliverables. Following this approach, the PPLM model ultimately contains (1) the activities, which are processes at various detail levels from the entire project down to the task level, required for completing the product, (2) the deliverables created during the project execution process, and (3) the various resources required for the project execution. Embedding this information consistently in the same unifying PPLM model yields all the structural relations—any relevant relations between two objects, and the procedural relations—any relevant relations between an object and a process in the system model. Abstraction provides for aggregating processes in nodes while creating the model in a hierarchical manner, catering to the human limited channel capacity (Mayer, 2001) and maintaining Miller's rule of 7±2 (Miller, 1956).

3.3 Modeling the UAV OPM-Based Project Plan

Following the basic OPM modeling guideline, the top-level process in the System Diagram, SD, (Fig. 2), **UAV Prototype Developing & Integrating**, represents the entire project process. It is assigned a maximum duration of 90 weeks. Three agents (human resources) carry out this process: **NMA Agents Set**, **Government Agents Set**, and **ECC Agents Set**. **Engine** and **GFE Payload** are parts of **UAV Prototype**, the deliverable of the project.

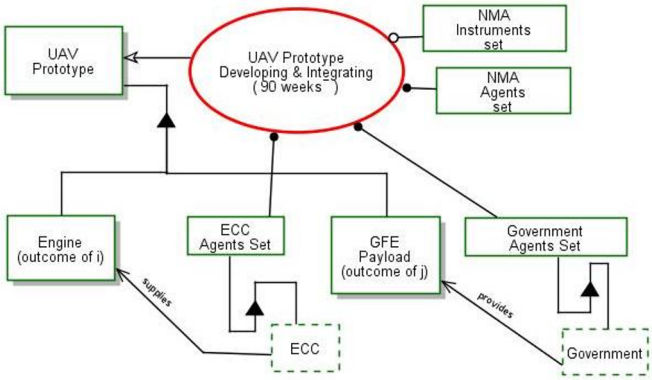

Fig. 2. OPD of the top-level System Diagram (SD) of UAV Prototype Developing & Integrating.

When creating the SD using OPCAT, equivalent natural language sentences, called Object Process Language (OPL) paragraph, are automatically generated, as given in Fig. 3.

UAV Prototype is physical.
UAV Prototype consists of **Engine (outcome of i)** and **GFE Payload (outcome of j)**.
    Engine (outcome of i) is physical.
    ECC supplies **Engine (outcome of i)**
    Engine (outcome of i) plays the role of **component**.
    GFE Payload (outcome of j) is physical.
    Government provides **GFE Payload (outcome of j)**
    GFE Payload (outcome of j) plays the role of **component**.
Government is environmental.
Government consists of **Government Agent Facilities Set**.
    Government Agent Facilities Set handles **UAV Prototype Developing & Integrating**.
ECC is environmental.
ECC consists of **ECC Agent Facilities Set**.
    ECC Agent Facilities Set handles **UAV Prototype Developing & Integrating**.
NMA Agent Facilities Set handles **UAV Prototype Developing & Integrating**.
UAV Prototype Developing & Integrating requires **NMA Instrument Facilities Set**.
UAV Prototype Developing & Integrating yields **UAV Prototype**.

Fig. 3. The automatically-generated OPL paragraph of the OPD in Fig. 2.

Fig. 4 presents SD1—the second-level OPD of the model, in which **UAV Prototype Developing & Integrating** is detailed using the in-zooming refinement mechanism that is built into OPM. When creating the SD using OPCAT, equivalent natural language sentences, called Object Process Language (OPL) paragraph, are automatically generated. At each detail level, the execution order of sub-processes within the in-zoomed process starts at the top of that process and proceeds downward, since the time line in an OPM model is vertical, flowing from top to bottom.

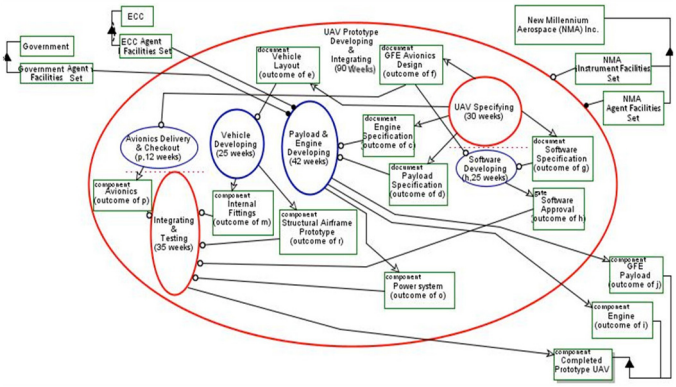

Fig. 4. OPD of the UAV Prototype Developing & Integrating In-zoomed

Some of the processes in SD1 (Fig. 4), e.g., **Avionics Delivery & Checkout** and **Software Developing**, are already tasks, i.e., simple leaf-level, atomic processes, as indicated by their thin ellipse contours. Other processes, indicated by thick ellipse contours, are non-simple and are further refined in subsequent lower-level OPDs.
In Fig. 4, there are two pairs of processes modelled with FS relationships (denoted by a dashed line between ellipses): **Software Developing** immediately follows **UAV Specifying** and **Integrating & Testing** immediately follows **Avionics Delivery & Checkout**. Two processes in the same figure are modelled using a PPLM Floating Start & Finish (FSF) relationship: The timing of **Vehicle Developing** floats within the timing of **Payload and Engine Developing.**
The first process in the sequence of **UAV Prototype Development & Integrating** is **UAV Specifying**. When this process ends, **Software Developing** can start, as it is linked with Finish-to-Start relationship one of the outcomes of the **UAV Specifying** process is the **Software Specification** object. **Software Specification** is assigned the role of document. This document is instrument to the **Software Developing** process. Therefore, **Software Developing** cannot start before this deliverable is generated by the **UAV Specifying** process. The outcome of **Software Developing** is **Software Approval**, an object which is assigned the role of a gate. This gate is instrument of the **Integrating & Testing** process. Therefore, **Integrating & Testing** cannot start before this deliverable is generated by **Software Developing**.

The **Payload & Engine Developing** process can start following the start of **UAV Specifying**, provided that the **Engine Specification** object has been created by the **Engine Specifying** process. Similarly, **Payload Specification** is an outcome of **Payload Specifying**. For the **Payload and Engine Developing** process to start, **ECC Agent Facilities Set** and **Government Agent Facilities Set** must also be available.

When the **Avionics Delivery & Checkout** process ends, the **Integrating & Testing** process can start, as they are linked with Finish-to-Start relationship. Starting the **Integrating & Testing** process requires also that the objects linked to it as instruments be available. These include the four components **Avionics**, **Internal Fittings Set**, **Structural Airframe Prototype**, and **Power System**, as well as one gate, **Software Approval**. **Integrating & Testing** is the last process within **UAV Prototype Developing & Integrating**. When it ends, **UAV Prototype** is generated, which contains **GFE Payload** and **Engine** outcomes of **Payload & Engine Developing**.

The abstraction of the original 23 processes (Fig. 1) into aggregating "parent" processes was based on planning practice. Each parent process is a logical cluster in the hierarchy of the project plan model. The processes in each cluster appear in the same OPD, which is a node in the OPD set tree. Different planners would probably suggest somewhat different hierarchical process decomposition (which is equivalent to task clustering). Similarly, different decompositions of hammocks would be defined by different planners using a Gantt chart. The point to note is that the "real" model is the flattened, all-inclusive model at the most detailed level. Clustering into logical groups helps humans, who are subject to the limited channel capacity, grasp the "big picture". This decomposition is not necessarily identical in all the models of the same system, but as long as the original 23 processes contained in the original UAV specification are identical and their dependence relationships are maintained the same, all these decompositions are acceptable. This is so since constructing all the relationships between the 23 leaf-level processes and the objects to fully account for the textual specification in Fig. 1 captures the entire project specification.

Since all the entities and their relations are represented in the project plan model, automatic procedures can be devised to generate familiar project views, including Gantt chart, Activities Network Plan, Critical Path, and a Work Breakdown Structure. In what follows, we present the potential for automatic extraction of the Work Breakdown Structure (WBS) planning representation from the OPM model and provide new OPM model-based representations.

4. CLASSICAL WBS AND OPM-BASED WBS

The Work Breakdown Structure (WBS) was initially developed as a product-oriented family tree by the United States Department of Defense (DoD), which in 1968 issued the "Work Breakdown Structures for Defense Materiel Items" (MIL-STD-881). WBS captures the work content within a project's scope in a hierarchical structure, and is usually accompanied with a dictionary describing each WBS element. The development of the WBS model involves breaking the overall work involved in a project into progressively smaller pieces down to the level of "work packages". These work packages are the elements for which required resources, budget, and duration can be estimated with relative confidence. The classical WBS model is usually developed before identifying the dependencies between activities and estimating their durations.

Fig. 5 and Fig. 6 show automatically extracted process WBS diagrams for two intermediate-level processes using the OPM unfolding mechanism. The first process WBS is for the **UAV Specifying** process (Fig. 5), and the second—for the **Integrating & Testing** (Fig. 6). They look like common hierarchical WBS views, except that the nodes, which are OPM processes, are denoted by ellipses instead of the commonly used rectangles. In both WBS views, each ellipse represents a process in the hierarchy and it contains information regarding the process duration, allocated resources, and budget, if inserted into the model. The resources (not shown in these views for simplicity) are denoted by rectangles—objects that are related to the various processes, as members of the **Agents Set** for human resources or of the **Instruments Set** for all other types of resources, except for budget, which has its own dedicated object, as argued earlier.

While the process WBS may be sufficient for classical project management, the PPLM approach advocates the planning and control of the correspondence between the project processes and the product deliverables. To this end, we suggest a new PPLM view, Object WBS (OWBS), which adds the involved objects (deliverables) to the processes hierarchy of the conventional process WBS views. The OWBS view is valuable as it provides a clear picture of the deliverables and how they flow amongst the planned WBS process nodes; it enables the comprehension of dependencies between nodes through the WBS decomposition in terms of deliverables.

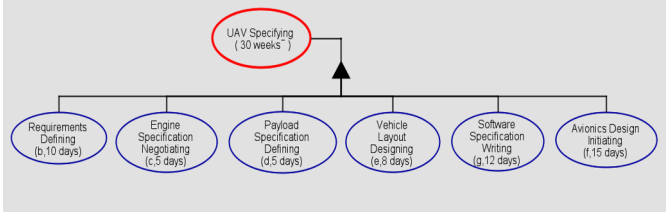

Fig. 5. An Automatically extracted process WBS for **UAV Specifying**.

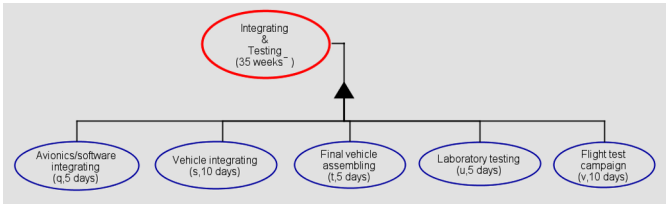

Fig. 6. An Automatically extracted process WBS for **Integrating & Testing**.

A partial OWBS view is presented in Fig.7 for the same **UAV Specifying** process shown in Fig. 5. A complete OWBS view is presented in Fig. 8 for the same **Integrating & Testing** process shown in Fig. 6. Both Fig. 7 and Fig. 8 contain only two levels since they were produced for nodes which are one level up from the task level, the lowest (leaf) level in the OPM-based UAV project plan model.

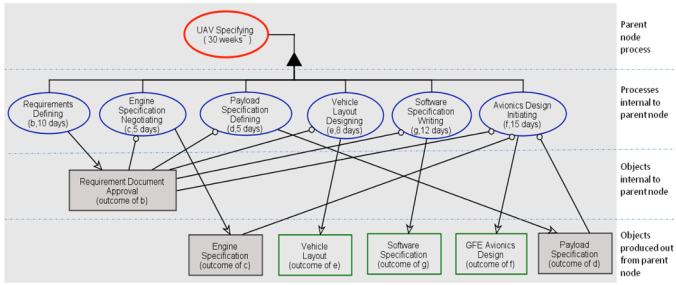

Fig. 7. An automatically extracted partial OWBS for **UAV Specifying**.

The OWBS view is constructed according to the following layout. The node for which the OWBS is extracted is placed at the top – this is the parent process, the top-level process in this hierarchy. *Internal processes* are processes that are contained in the in-zoomed parent process or are its parts. These internal processes are placed horizontally beneath the parent process and are connected with the parent by an aggregation-participation link (the solid triangle). Beneath each one of these internal processes are the exclusively internal deliverables of that process, along with their links to the process. An *exclusively internal deliverable* is a deliverable that is required only between internal processes. All the other deliverables are external—they are (also or only) required by processes external to the OWBS. External deliverables are placed at the bottom of the OWBS view.

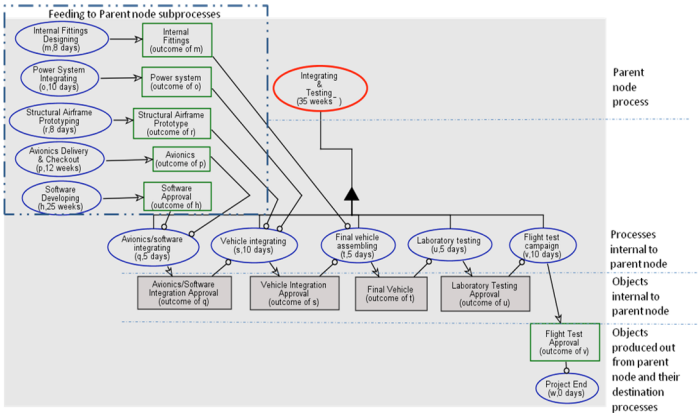

Fig. 8. An automatically extracted partial OWBS for **Integrating & Testing**

A deliverable of an OWBS can be a *mixed deliverable*—a deliverable to at least two processes, of which at least one is internal and another—external. Like internal deliverables, mixed deliverables are solid, but they are placed at the bottom of the OWBS view, along with the external deliverables. We demonstrate this by the six deliverables produced by subprocesses of the **UAV Specifying** node in the OWBS at the top of Fig. 7: **Requirements Document Approval**, **Vehicle Layout**, **Software Specification**, **GFE Avionics Design**, **Payload Specification**, and **Engine Specification**. **Requirements Document Approval** is an internal deliverable, as it is generated and required only by internal processes—subprocesses of **UAV Specifying**. **Requirements Document Approval** is therefore marked by a solid rectangle and placed at the first level under the node's subprocesses. **Engine Specification** and **Payload Specification** are mixed deliverables; they are required by internal processes, but based on the PPLM model, they are also inputs to external processes. Being mixed variables, **Engine Specification** and **Payload Specification** are solid rectangles, but they are placed at the bottom of the OWBS view together with the remaining three deliverables, **Vehicle Layout**, **Software Specification**, and **GFE Avionics Design**, which are external, as indicated by their blank rectangles.

4. CONCLUSIONS

Model-based project planning is part of the joint Project-Product Lifecycle Management (PPLM) approach. It provides the ability to gain deep comprehension of the project-product super-system and to derive various common project management views from the unifying PPLM model. The joint project-product OPM-based model combines the project's activities and timing with the product's structure in a single conceptual model.

The newly suggested OWBS view is valuable for all stake holders of the project plan since it provides a clear picture of the deliverables and how they flow amongst the planned WBS process nodes; it enables the comprehension of dependencies between nodes through the WBS decomposition in terms of deliverables.

The various consistent views extracted from the model-based project model enable the project manager to focus on advancing the completion of the required deliverables rather than on performing processes that are not necessarily linked to anticipated outcomes.

REFERENCES

Cook S.C., Kasser J.E., and Ferris T.K.J., "Elements of a Framework for the Engineering of Complex Systems, "in *Proc. 9th ANZSYS Conference, Systems in Action*, Melbourne, Australia, 2003, Paper 3000079.

de Weck O. and Lyneis J.M., *MIT ESD.36 System Project Management* course assignments, 2008.

Dori D., *Object-Process Methodology – A Holistic Systems Paradigm*. Springer Verlag, Berlin, Heidelberg, New York, 2002.

Dori D., Reinhartz-Berger I., and Sturm A., Developing Complex Systems with Object-Process Methodology using OPCAT. Industrial Presentation in Proc. 22nd International Conference on Conceptual Modeling (ER 2003), Chicago Illinois, October 13-16, 2003. http://www.opcat.com/

Eppinger S. and Salminen V., "Patterns of Product Development Interactions, " in *Proc. International Conference On Engineering Design, ICED01*, Glasgow, August 21-23, 2001, pp 283-290.

Fischer E.D. (Ed), *Workflow Handbook*. Fes Inc., Lighthouse Point, Florida, USA, 2005.

Levy K.F., Thompson G.L. and Wiest J.D. The ABCs of the Critical Path Method, *Harvard Business Review, #63508, Harvard Business School Publishing*, 1963.

Mayer R.E., Multimedia Learning. Cambridge University Press. 2001.

Miller G., The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, vol. 63, pp. 81-9. 1956.

PMBoK, Project Management Institute, 2008.

Sharon A., Perelman V., and Dori D., A Project-Product Lifecycle Management Approach For Improved Systems Engineering Practices. *Proc. INCOSE 18th Annual International Symposium, 6th Biennial European Systems Engineering Conference,* Utrecht, the Netherlands, 2008.

Sharon A., Dori D., and de Weck O., "Is there a Complete Project Plan? A Model-Based Project Planning Approach, "*Proc. 19th Annual International INCOSE Symposium*, Singapore, 2009.

Sharon A., Dori D., and de Weck O., Project Management vs. Systems Engineering Management: A Practitioners' View on Integrating the Project and Product Domains. Systems Engineering, 14(4), pp. 427-440, Oct. 2011.

Sim S.K. and Duffy A.H.B., Towards an Ontology of Generic Engineering Design Activities. *Research n Engineering Design*, vol. 14, pp. 200-223. 2003.

Work Breakdown Structures for Defense Material Items, Military Standard Mil-Std-881-1968.