ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τεχνολογία Λογισμικού, 7ο/9ο εξάμηνο 2018-2019

Τεχνολογία Λογισμικού

Ν.Παπασπύρου, Αν.Καθ. ΣΗΜΜΥ, nickie@softlab.ntua,gr

Β.Βεσκούκης, Αν.Καθ. ΣΑΤΜ, v.vescoukis@cs.ntua,gr

Κ.Σαΐδης, ΠΔ 407, saiko@softlab.ntua.gr

UML diagrams, consistency, facets, examples

Παραδοτέα εργασίας, revisited

Παραδοτέο	Ομάδα 5 ατόμων	Ομάδα 6 ατόμων	Ομάδα 7 ατόμων
Documentation - Diagrams			
Εγγραφο StRS - Stakeholders Requirements Specification	ΝΑΙ	ΝΑΙ	ΝΑΙ
Εγγραφο SRS - Software Requirements Specification	2-3 Use Cases	3-4 Use Cases	4-5 Use Cases
Διαγράμματα UML Use Case	ΝΑΙ	ΝΑΙ	ΝΑΙ
Διαγράμματα UML Activity	Αντίστοιχα με τα Use Cases		
Διαγράμματα UML Sequence ή Communication	Αντίστοιχα με τα Use Cases		
Διαγράμματα UML Deployment / Component	ΝΑΙ	ΝΑΙ	ΝΑΙ
Διαγράμματα UML Class	ΝΑΙ	ΝΑΙ	ΝΑΙ
Διαγράμματα ER	ΝΑΙ	ΝΑΙ	ΝΑΙ
Source code - implementation			
Πηγαίος κώδικας εφαρμογής - web client	ΝΑΙ	ΝΑΙ	ΝΑΙ
Πηγαίος κώδικας εφαρμογής - server	ΝΑΙ	ΝΑΙ	ΝΑΙ
RESTful API	ΝΑΙ	ΝΑΙ	ΝΑΙ
Mobile app		Προαιρετικό	Προαιρετικό
Εκτελέσιμη μορφή	ΝΑΙ	ΝΑΙ	ΝΑΙ

UML: a communication tool

What is communication about:

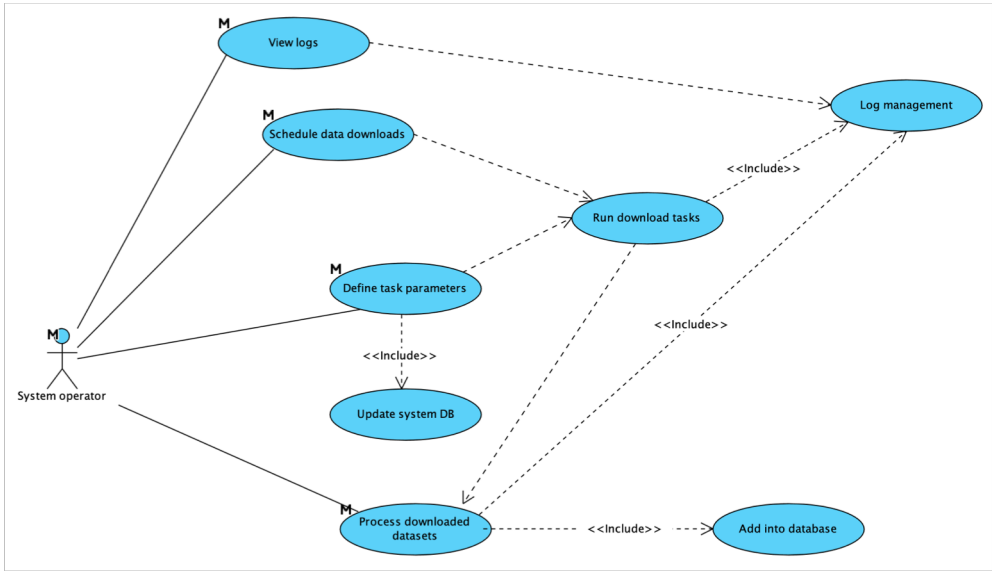
- Who communicates?
- What is communicated?
- To whom is it communicated?

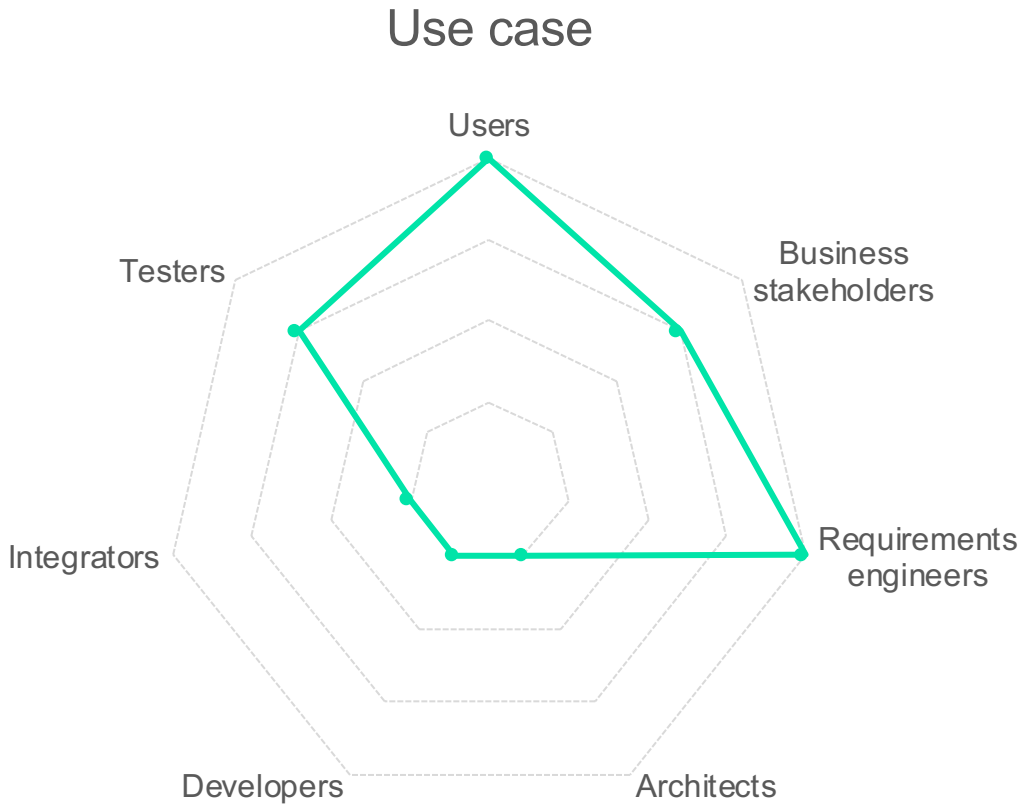
Challenges

- Not uniform (if any, at all) understanding of the means of communication
- Different perspectives
- Conflicting interests, mutually exclusive constraints and goals

UML diagrams may have different readings…

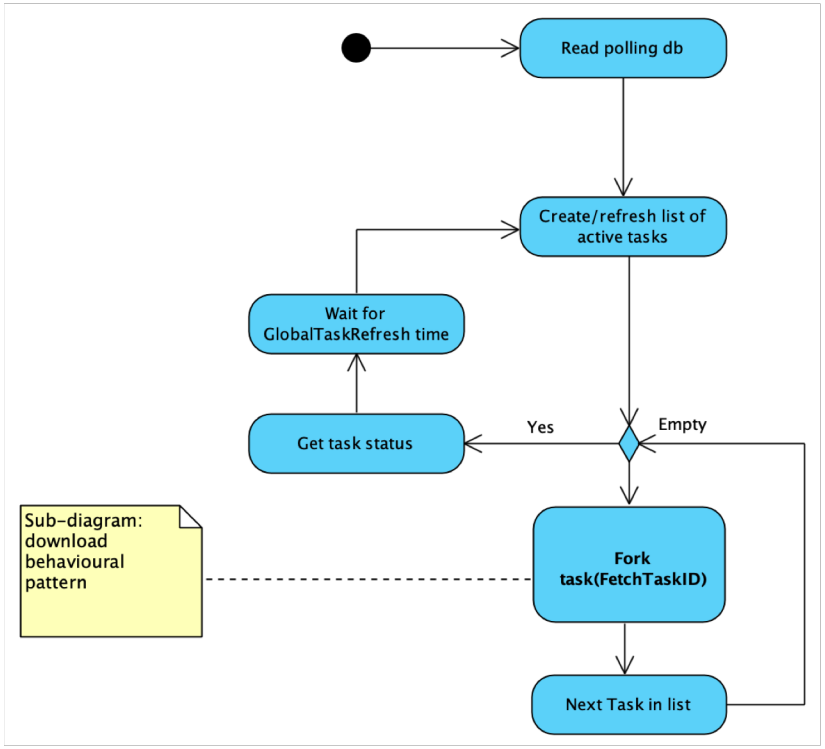
Use case diagrams communicate system's aspects visible to users

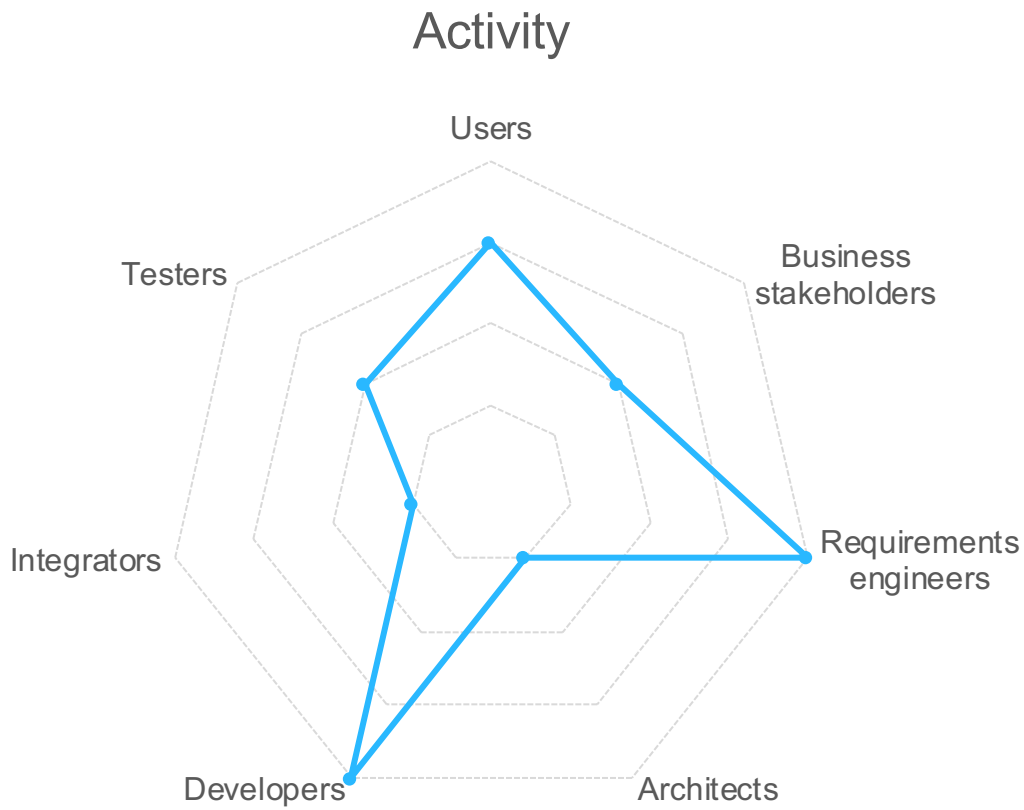




UML diagrams may have different readings…

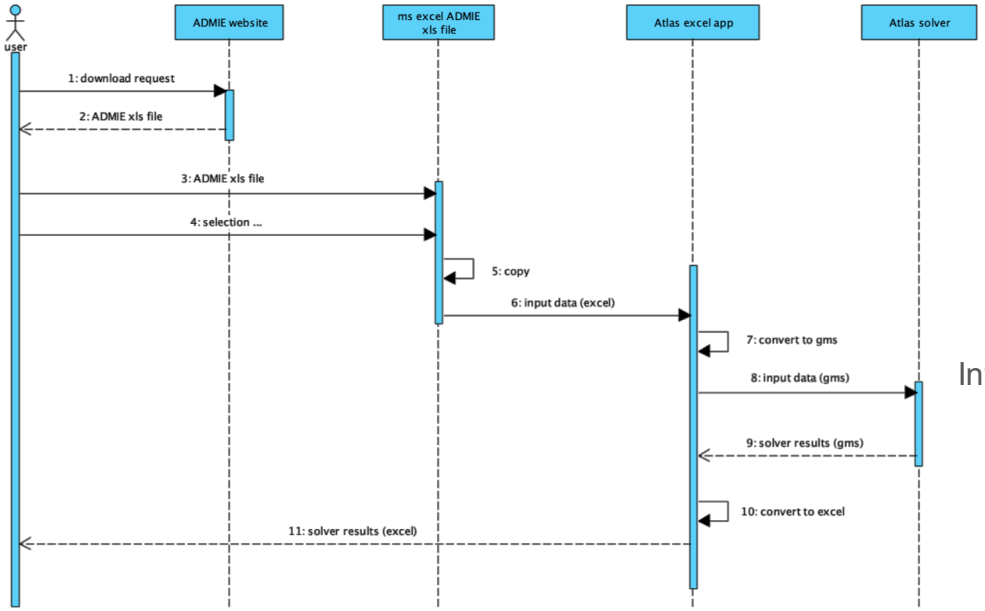
Activity diagrams communicate system's
(intended) behavior in a comprehensive, familiar form

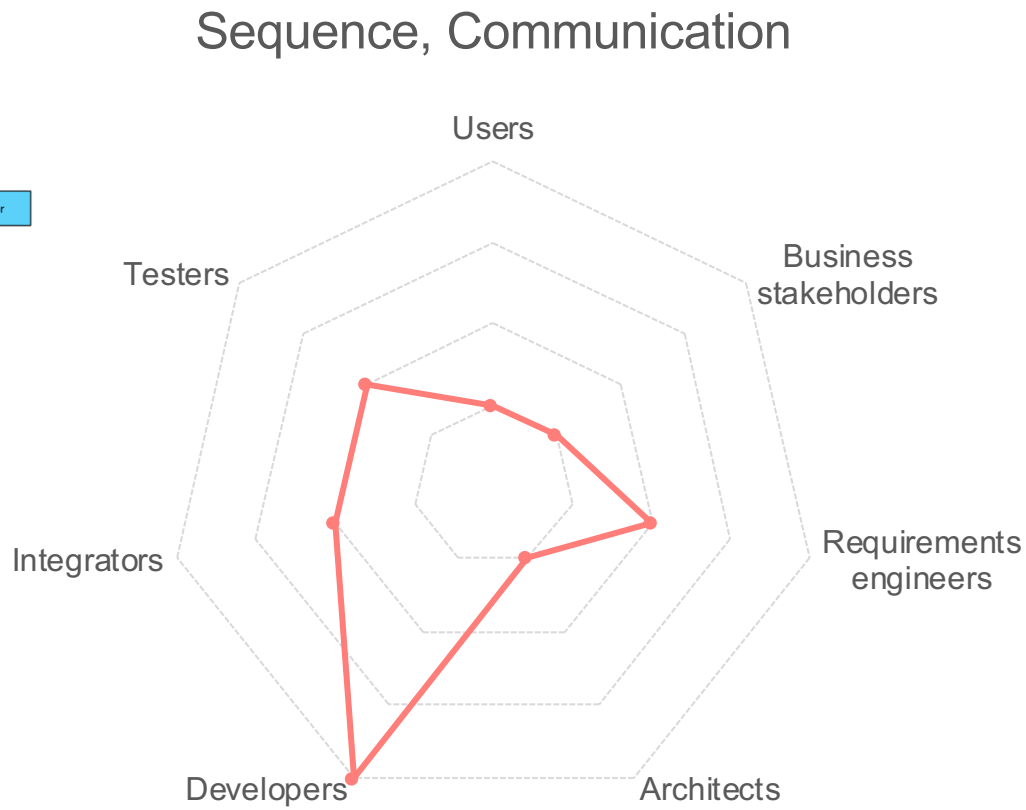




UML diagrams may have different readings…

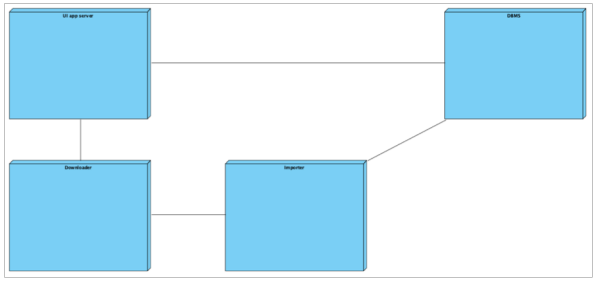
Sequence and communication (collaboration) diagrams communicate detailed interactions between system's components

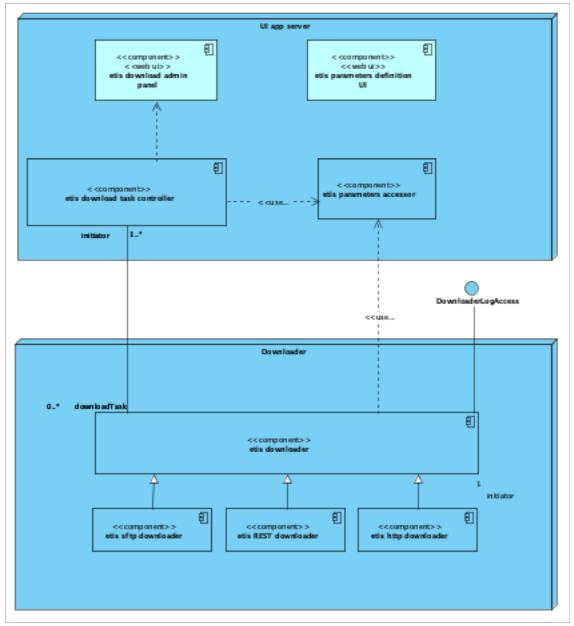


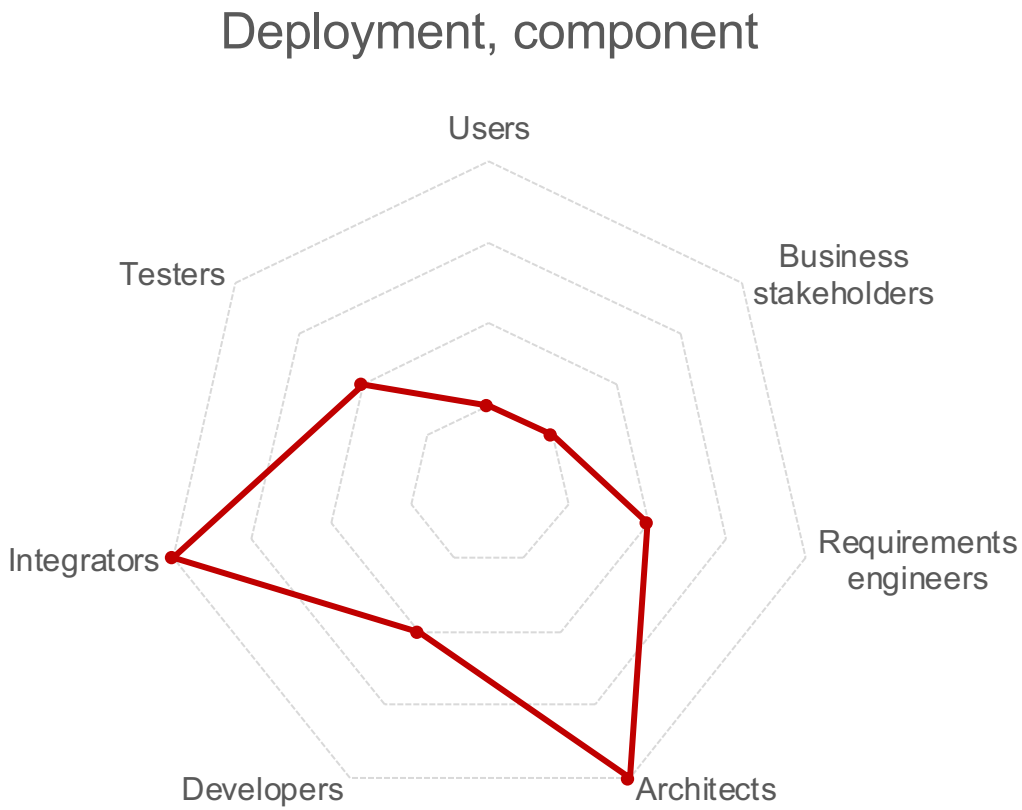


UML diagrams may have different readings…

Deployment and component diagrams communicate aspects of system's structure

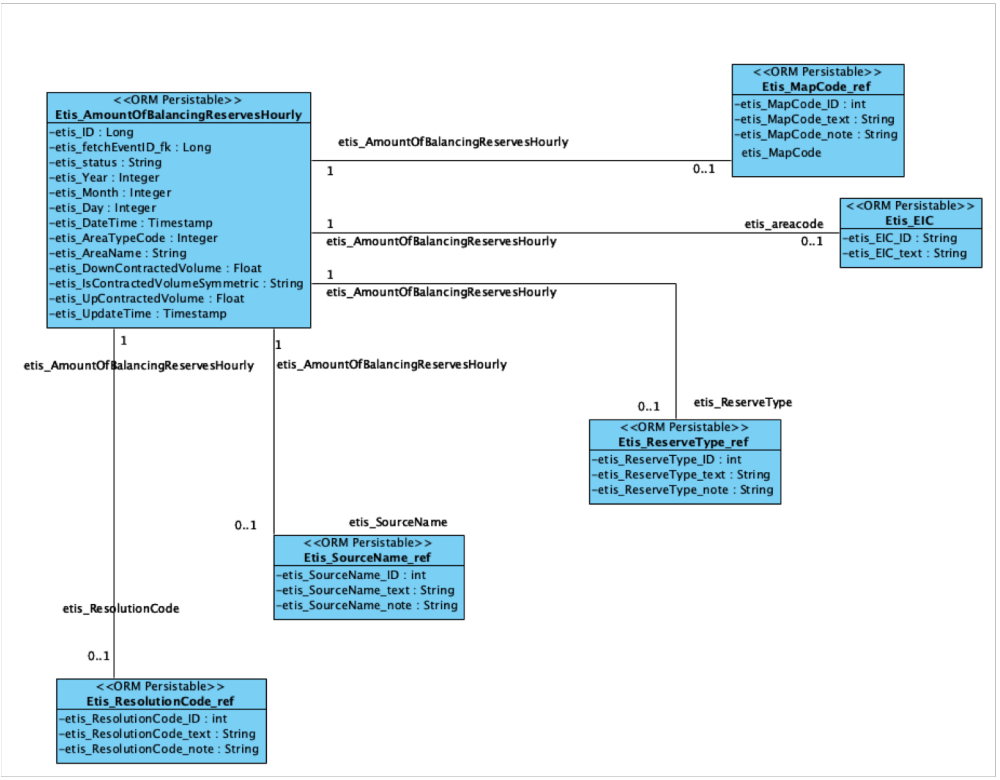


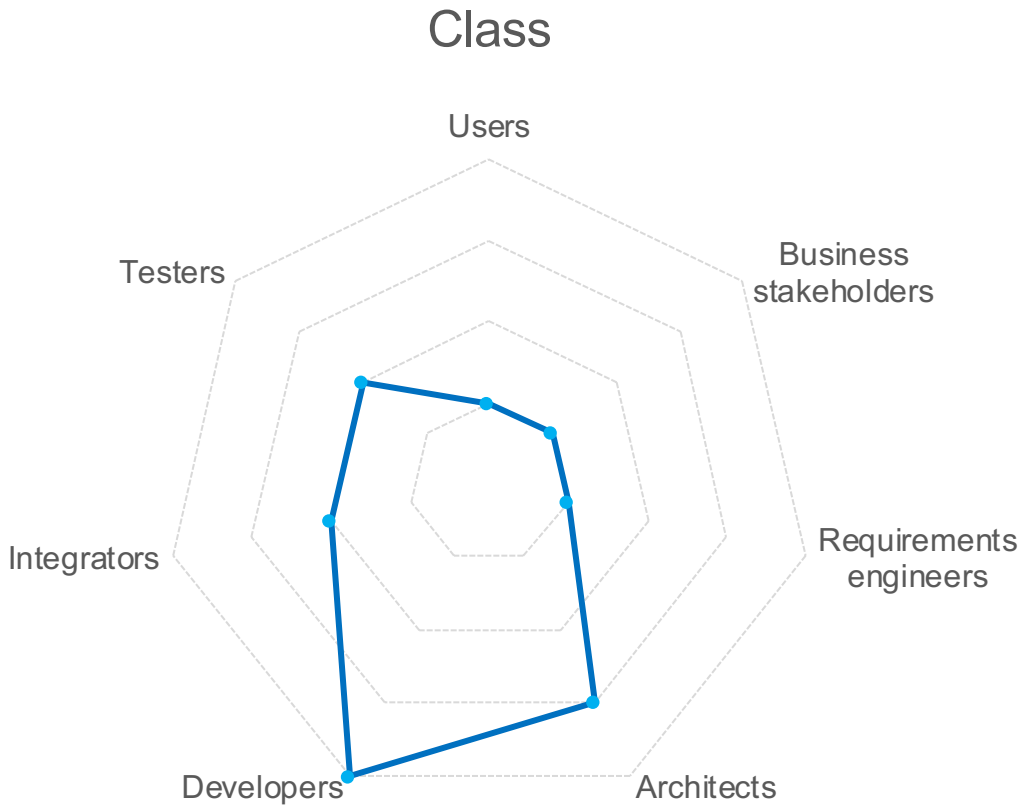




UML diagrams may have different readings…

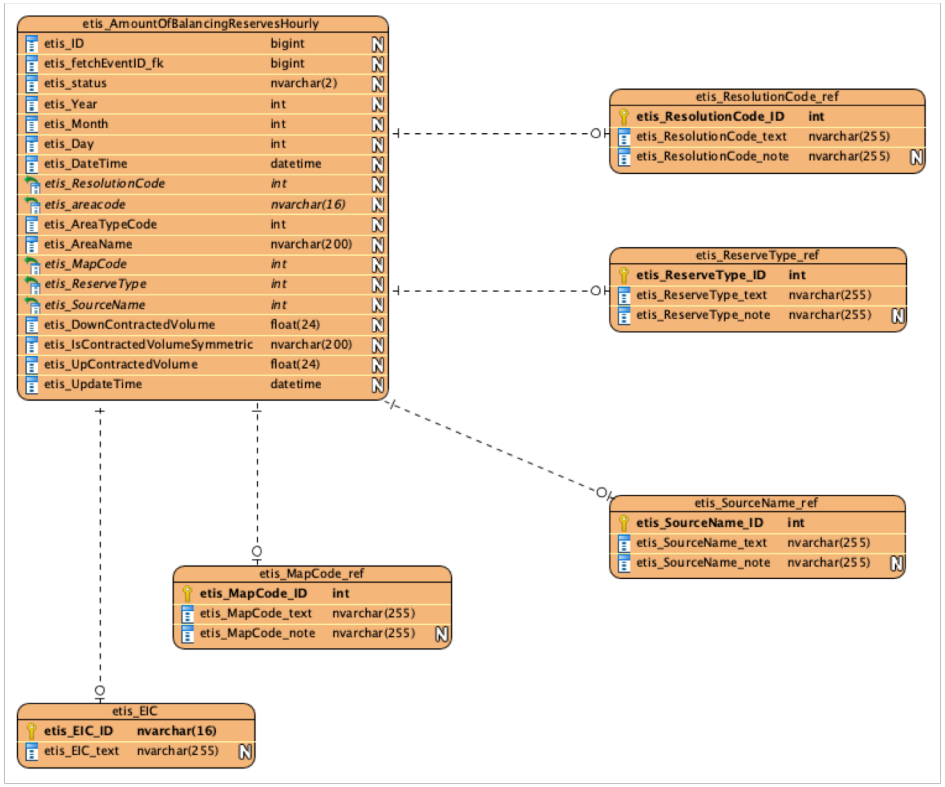
Class diagrams communicate implementation aspects at a high level of detail

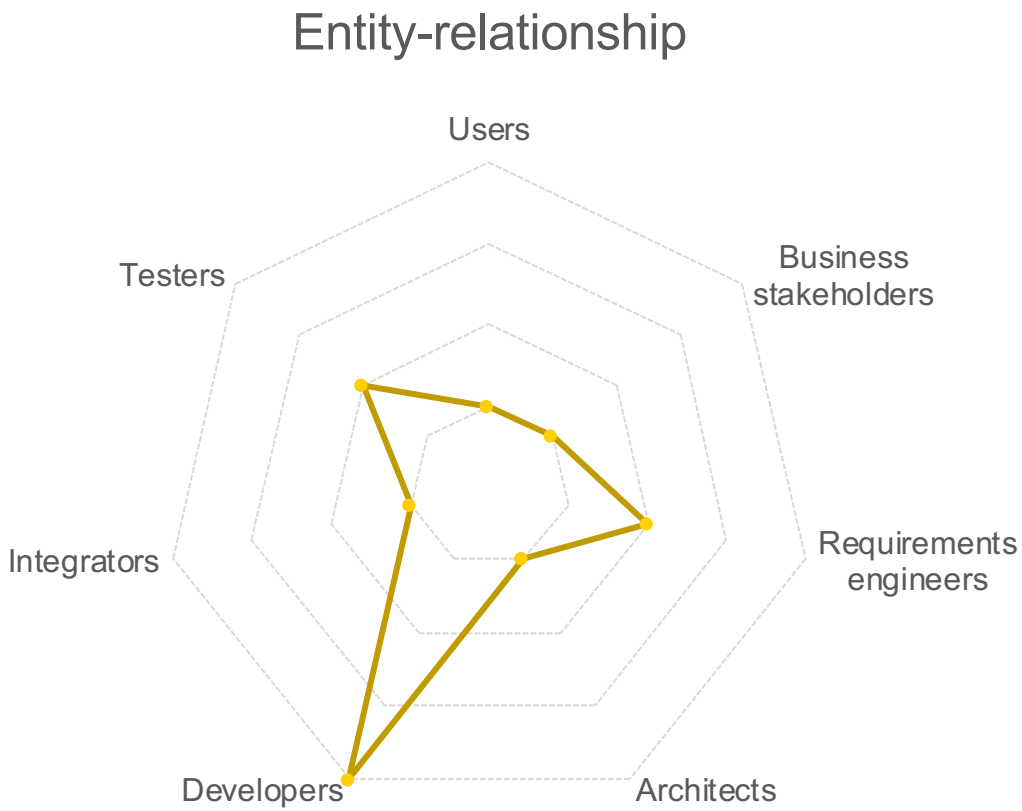




UML diagrams may have different readings…

Entity-Relationship diagrams communicate
data organization aspects (from semantic to DBMS implementation)





Possible communications in software engineering

Interactions (read from row to column)	Users	Business stakeholders	Requirements engineers	Architects	Developers	Integrators	Testers
Users		Provide insights	Negotiate features	Architects discuss on deployment options	Developers collaborate on implementation with Architects	Users provide insights to Business stakeholders	Collaborate on UAT
Business stakeholders	Propose features		Explain business case				
Requirements engineers	Confirm feature requests	Agree on product features		Elaborate on architecture	Discuss on implementation decisions		Align perspectives
Architects		Discuss strategies	Provide options		Discuss on implementation decisions	Discuss on deployment options	Provide test cases
Developers		Negotiate salaries ☺	Consult when needed	Collaborate on implementation		Provide feedback if needed	Receive feedback & tickets
Integrators		Discuss costing		Provide feedback	Provide support		
Testers	Propose UAT	Provide executive reports	Finalize UAT	Propose test cases	Define test cases	Propose test cases	

Possible communications in software engineering

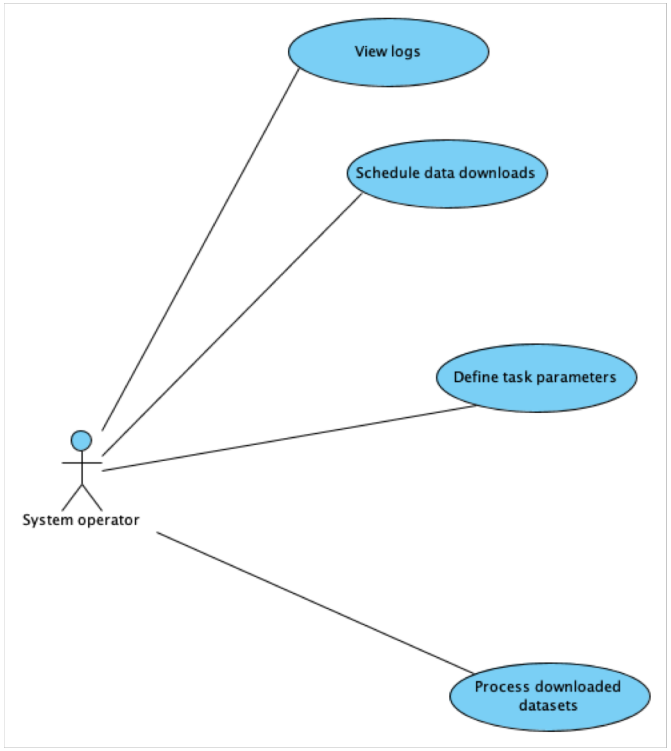
Interactions (read from row to column)	**Users**	**Business stakeholders**	**Requirements engineers**	**Architects**	**Developers**	**Integrators**	**Testers**
Users		Provide insights	Negotiate features				Collaborate on UAT
Business stakeholders	Propose features		Explain business case				
Requirements engineers	Confirm feature requests	Agree on product features		Elaborate on architecture	Discuss on implementation decisions		Align perspectives
Architects		Discuss strategies	Provide options		Discuss on implementation decisions	Discuss on deployment options	Provide test cases
Developers		Negotiate salaries ☺	Consult when needed	Collaborate on implementation		Provide feedback if needed	Receive feedback & tickets
Integrators		Discuss costing		Provide feedback	Provide support		
Testers	Propose UAT	Provide executive reports	Finalize UAT	Propose test cases	Define test cases	Propose test cases	

UML as a communication tool in software development

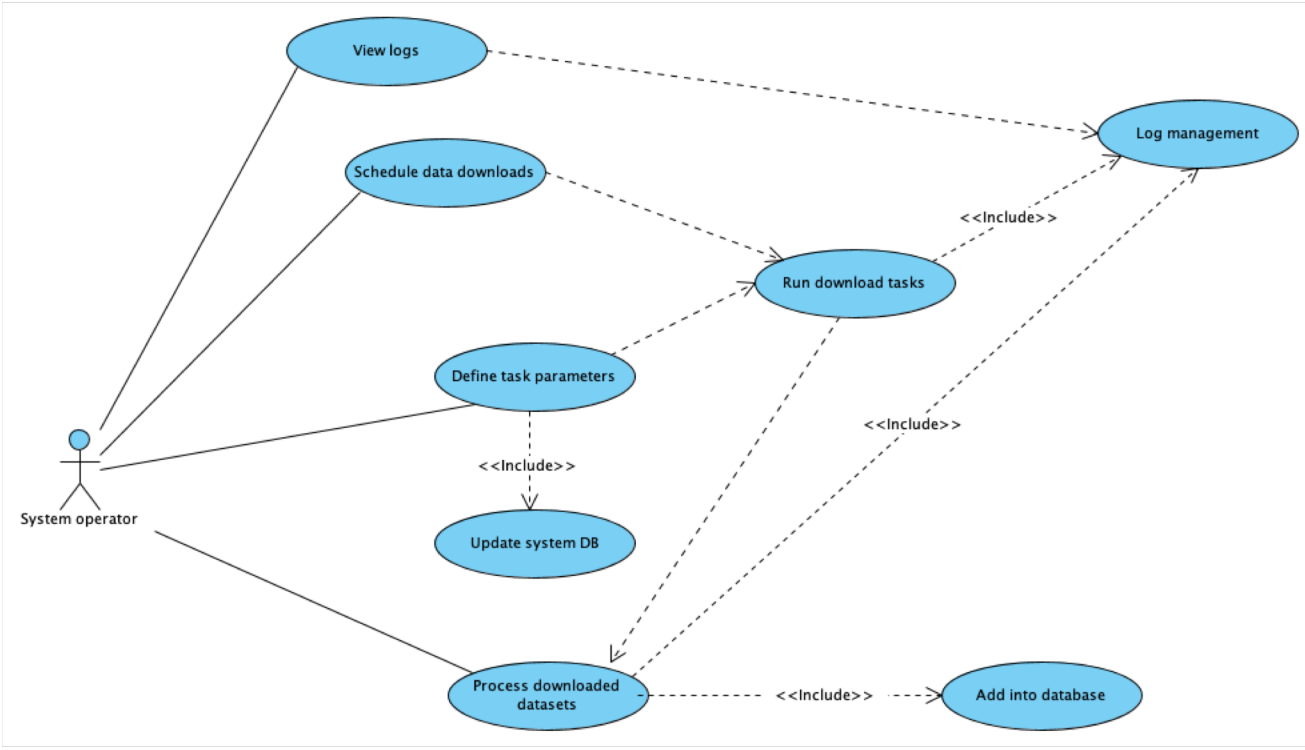
Interactions (read from row to column)	**Users**	**Business stakeholders**	**Requirements engineers**	**Architects**	**Developers**	**Integrators**	**Testers**
Users		Provide insights Use case	Negotiate features Activity, Use case				Collaborate on UAT Activity, Use case
Business stakeholders	Propose features Use case		Explain business case Use case				
Requirements engineers	Confirm feature requests Use case, activity	Agree on product features Use case, deployment		Elaborate on architecture Deployment, component	Discuss on implementation decisions Deployment, component, class		Align perspectives Deployment, component, class
Architects		Discuss strategies Deployment	Provide options Deployment, component		Discuss on implem-entation decisions Component, sequence, communication, ER	Discuss on deployment options Deployment, component	Provide test cases Component, deployment
Developers		Negotiate salaries ☺ No UML diagram (yet?)	Consult when needed Use case	Collaborate on implementation Component, sequence, communication, ER		Provide feedback if needed Component	Receive feedback & tickets Component, class, communication, sequence, ER
Integrators		Discuss costing Deployment		Provide feedback Deployment, component	Provide support Component		
Testers	Propose UAT Use case	Provide executive reports Deployment, use case	Finalize UAT Use case	Propose test cases Use case, deployment	Define test cases Component, class	Propose test cases Component, use case	

A real-life example: energy information system

A use case diagram…



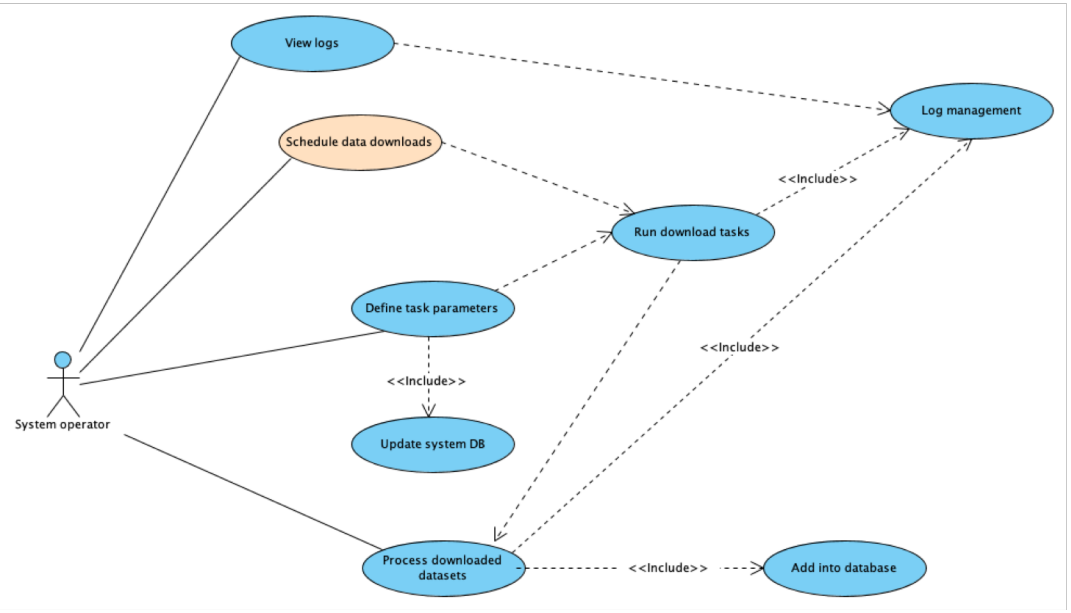
as users would see it…

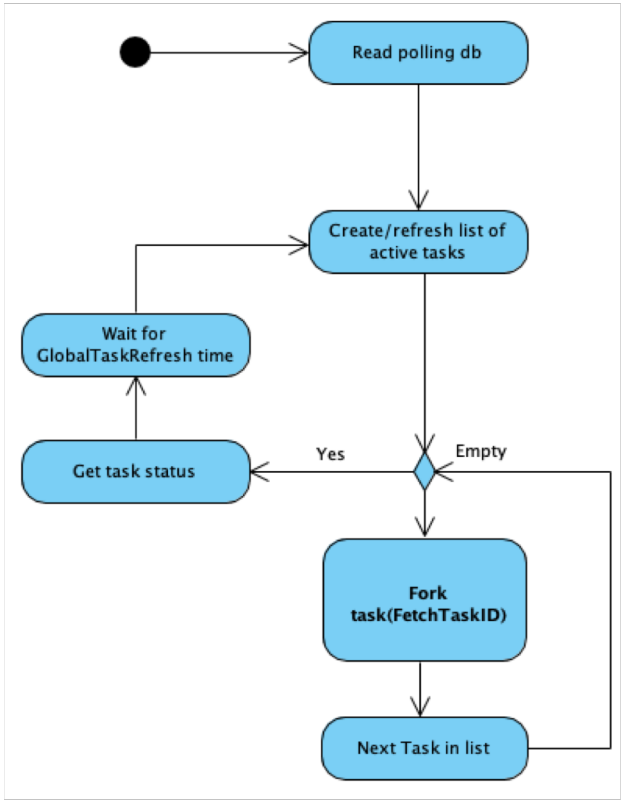


as requirement engineers would further work with it…

A real-life example: energy information system

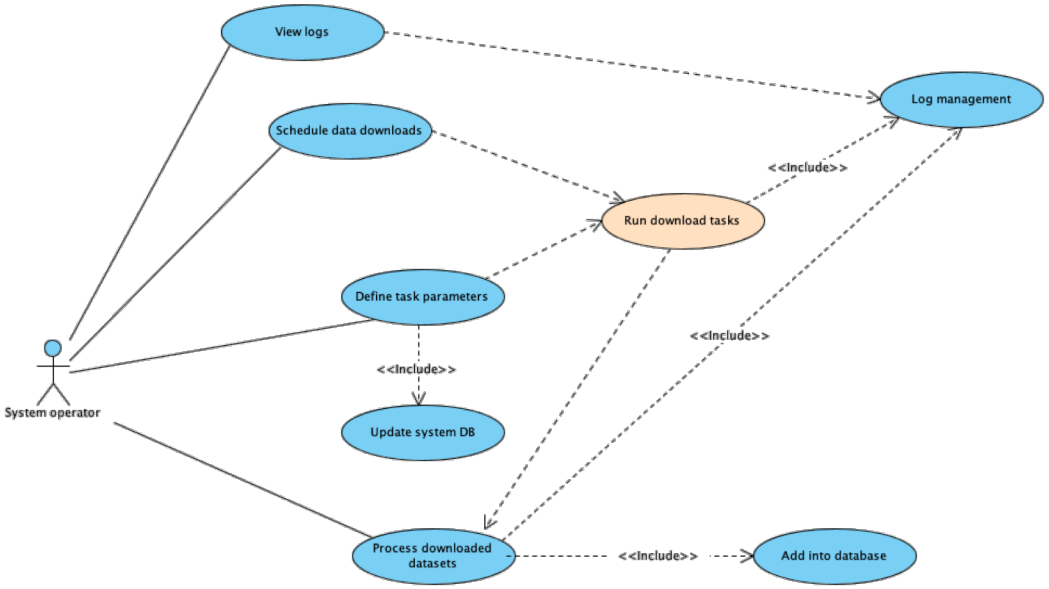
Description (specification) of the functionality in a Use Case by a UML activity diagram

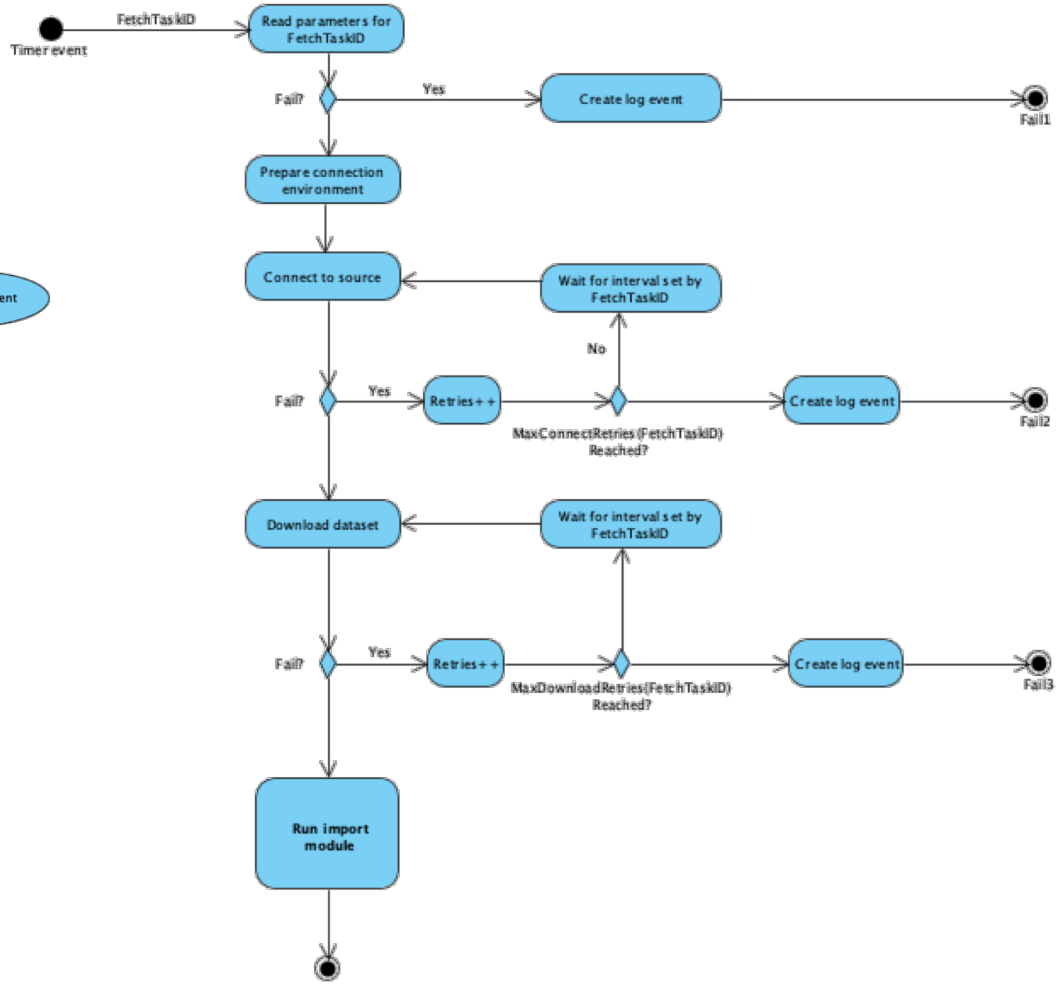




A real-life example: energy information system

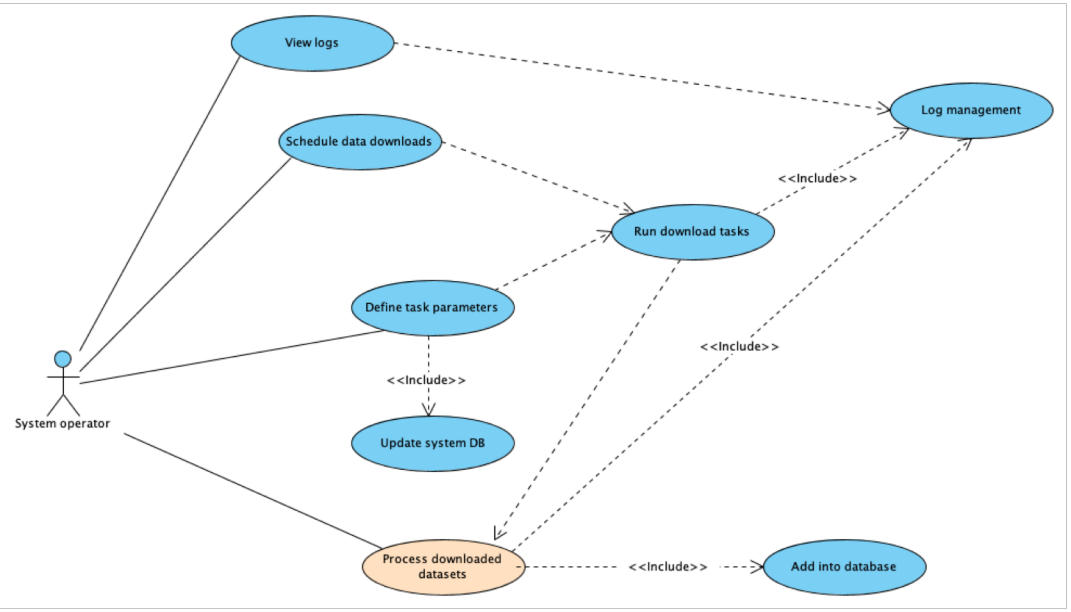
Description (specification) of the functionality in a Use Case by a UML activity diagram

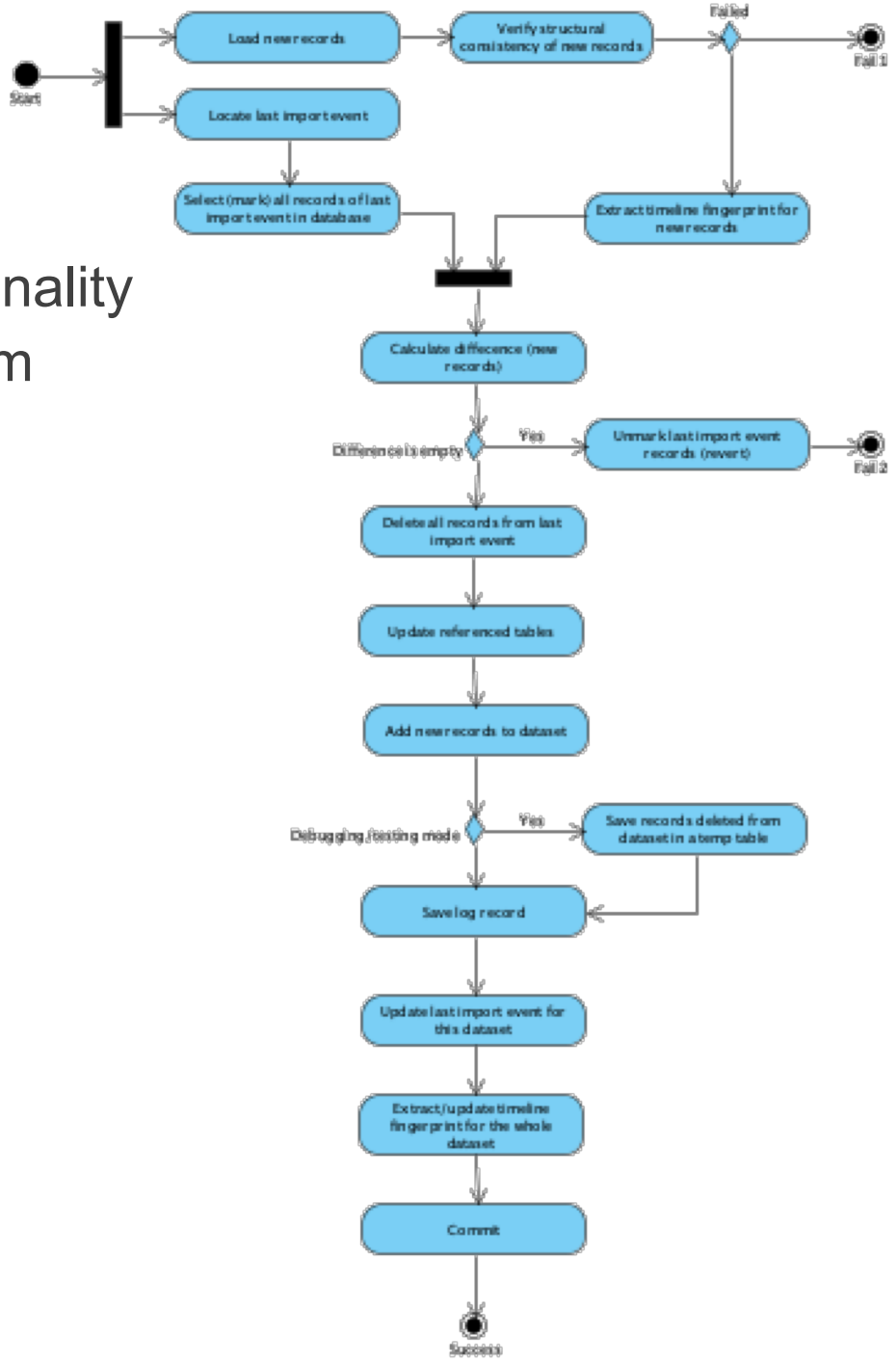




A real-life example: energy information system

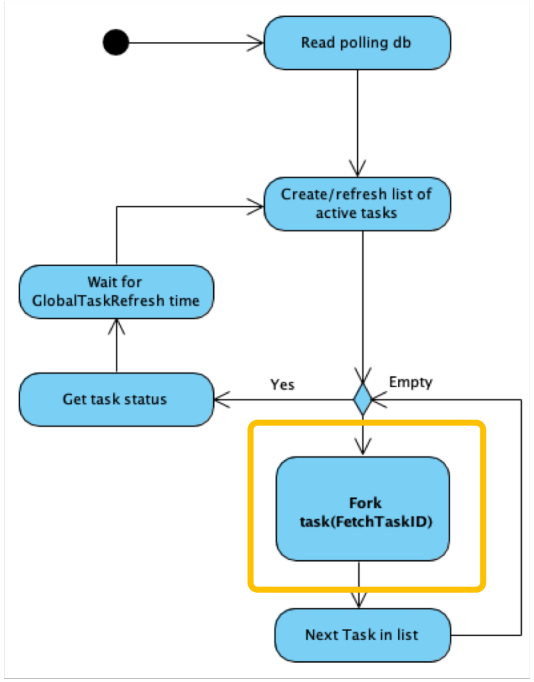
Description (specification) of the functionality in a Use Case by a UML activity diagram

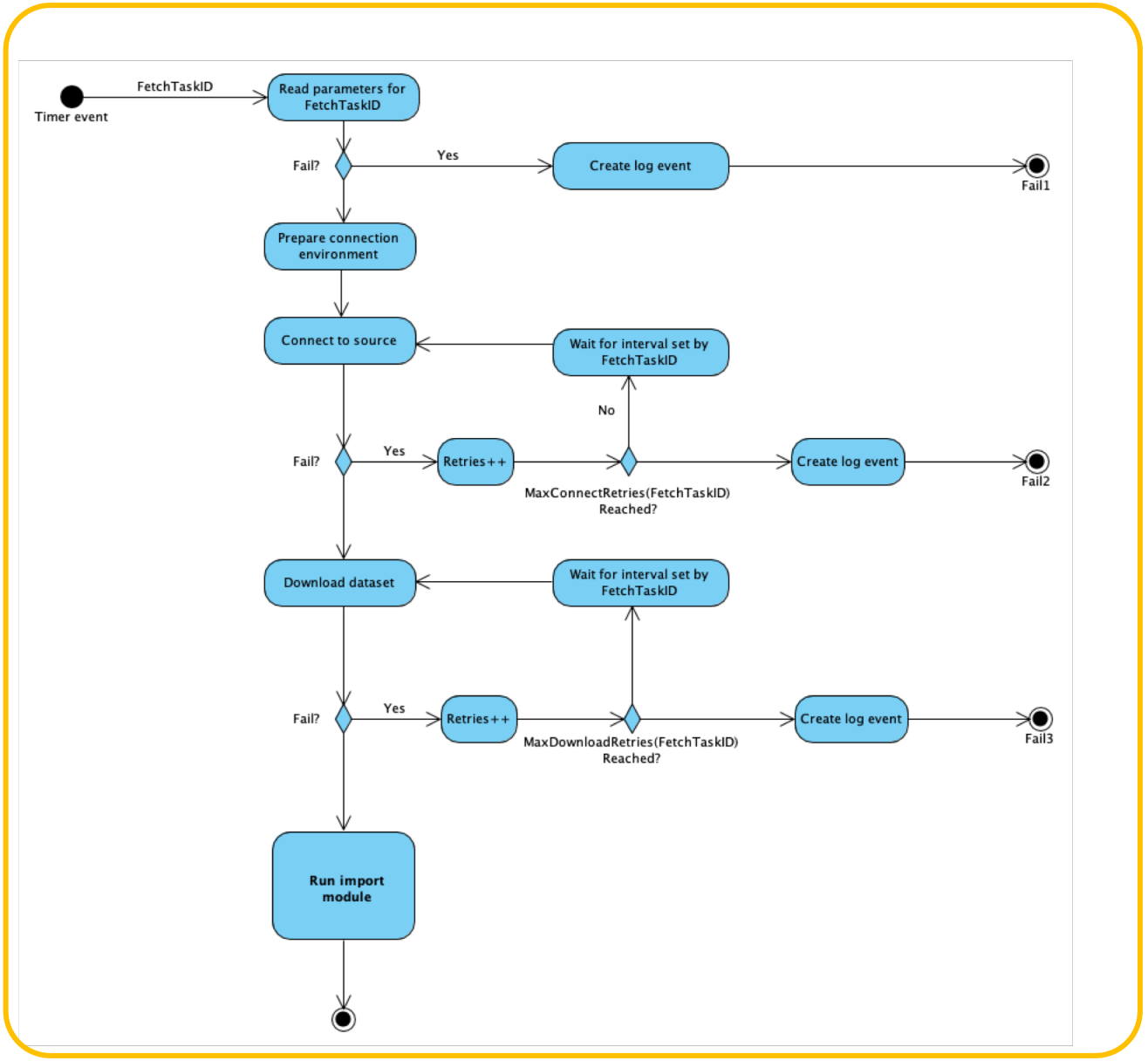




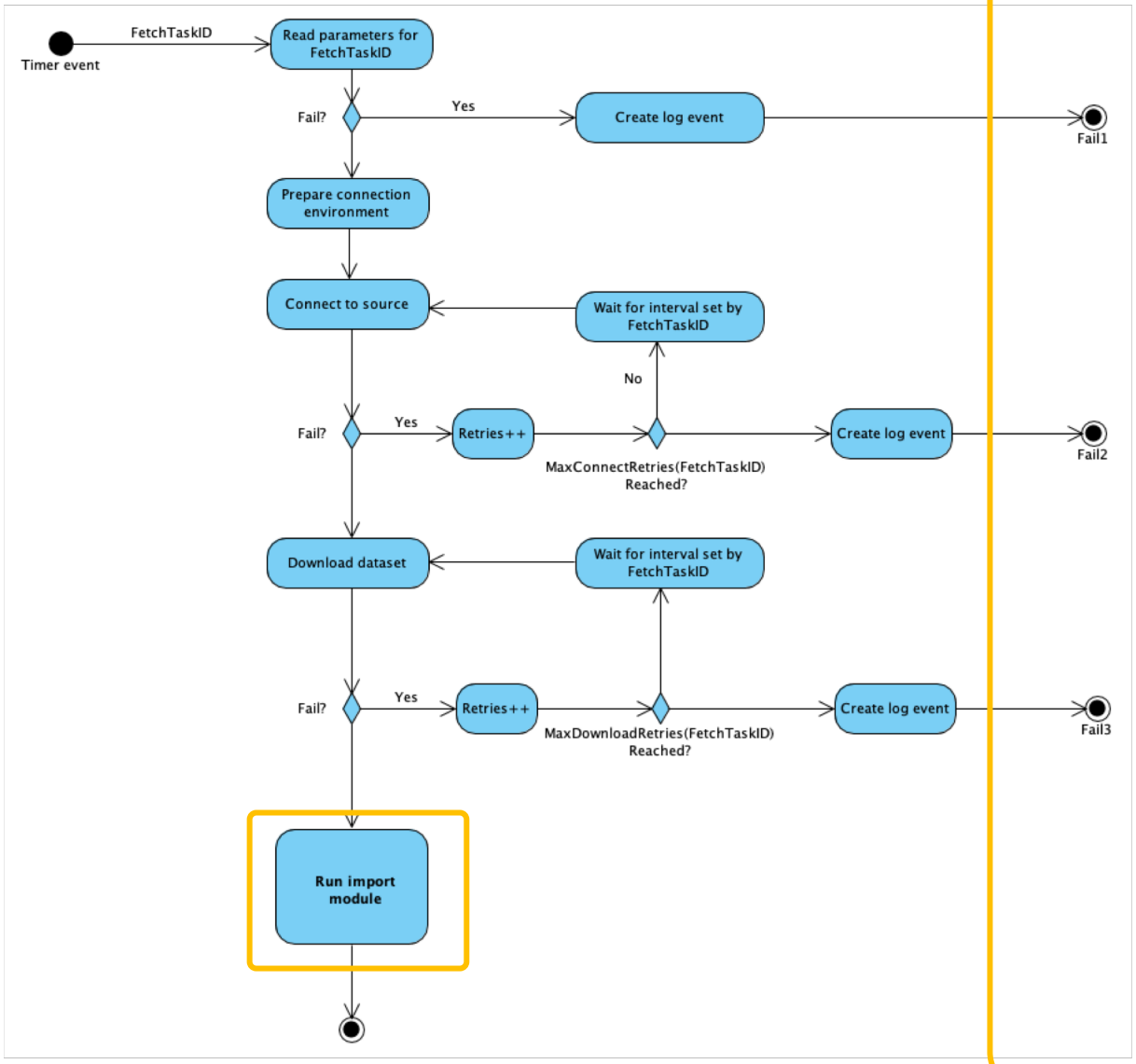
A real-life example: energy information system

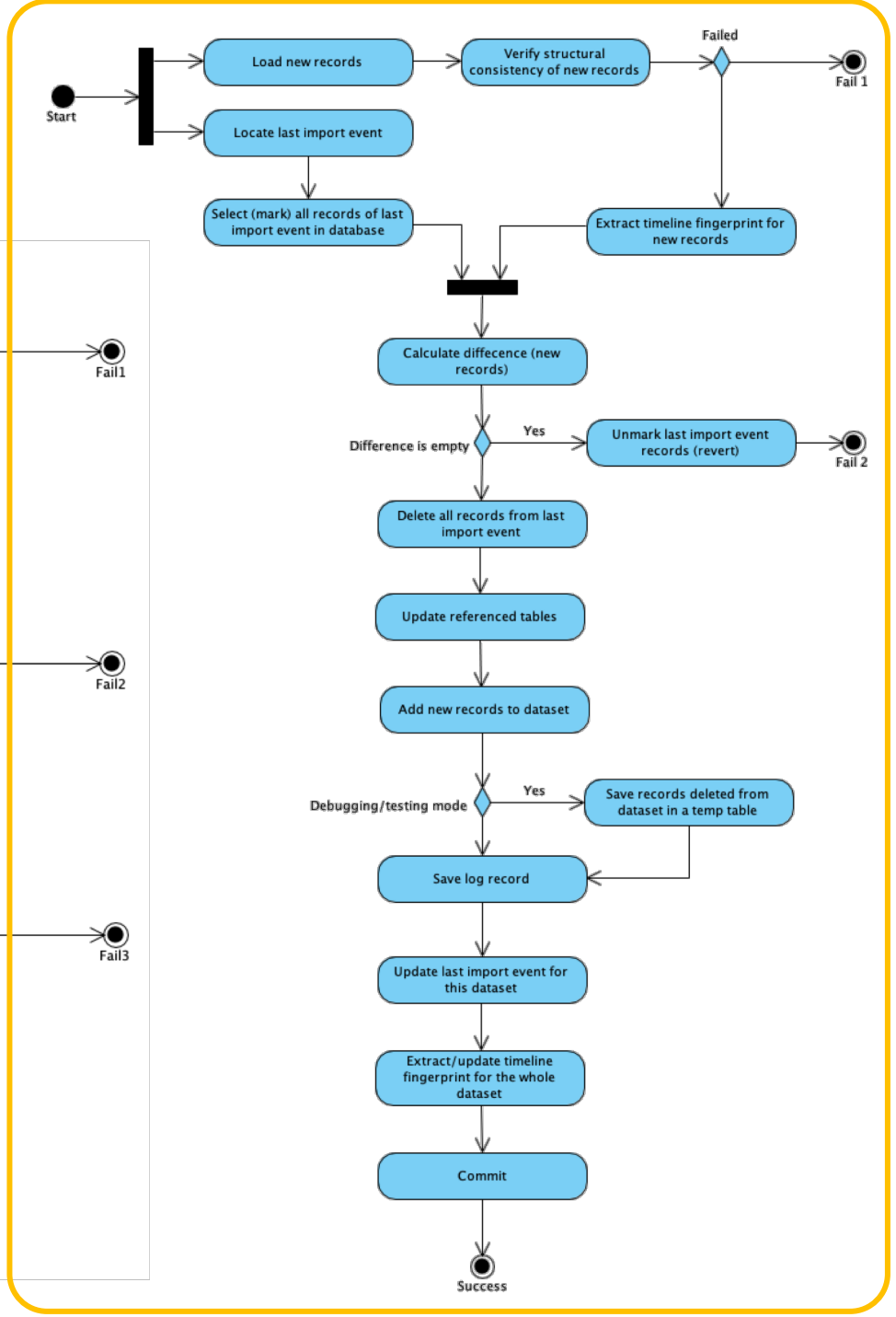
Use of sub diagrams





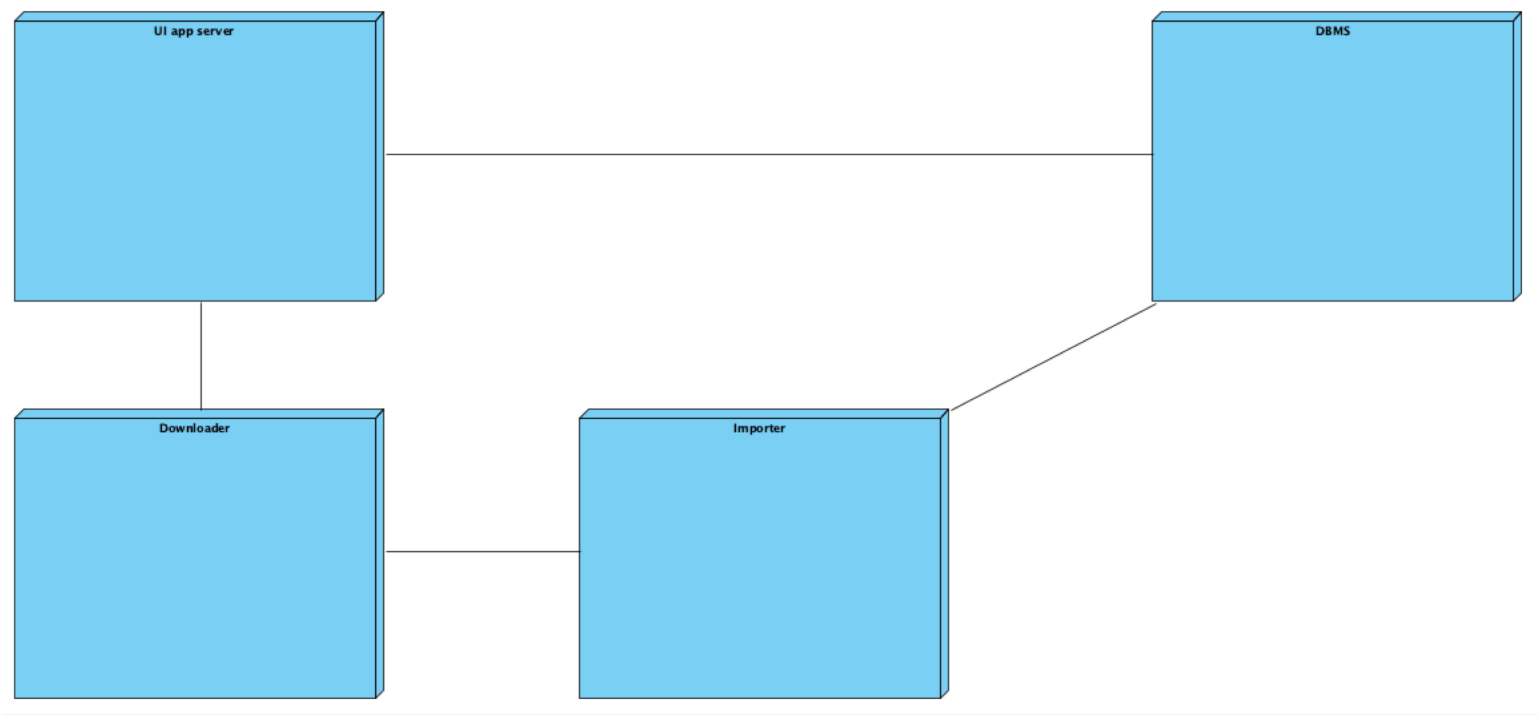
A real-life example: energy information system





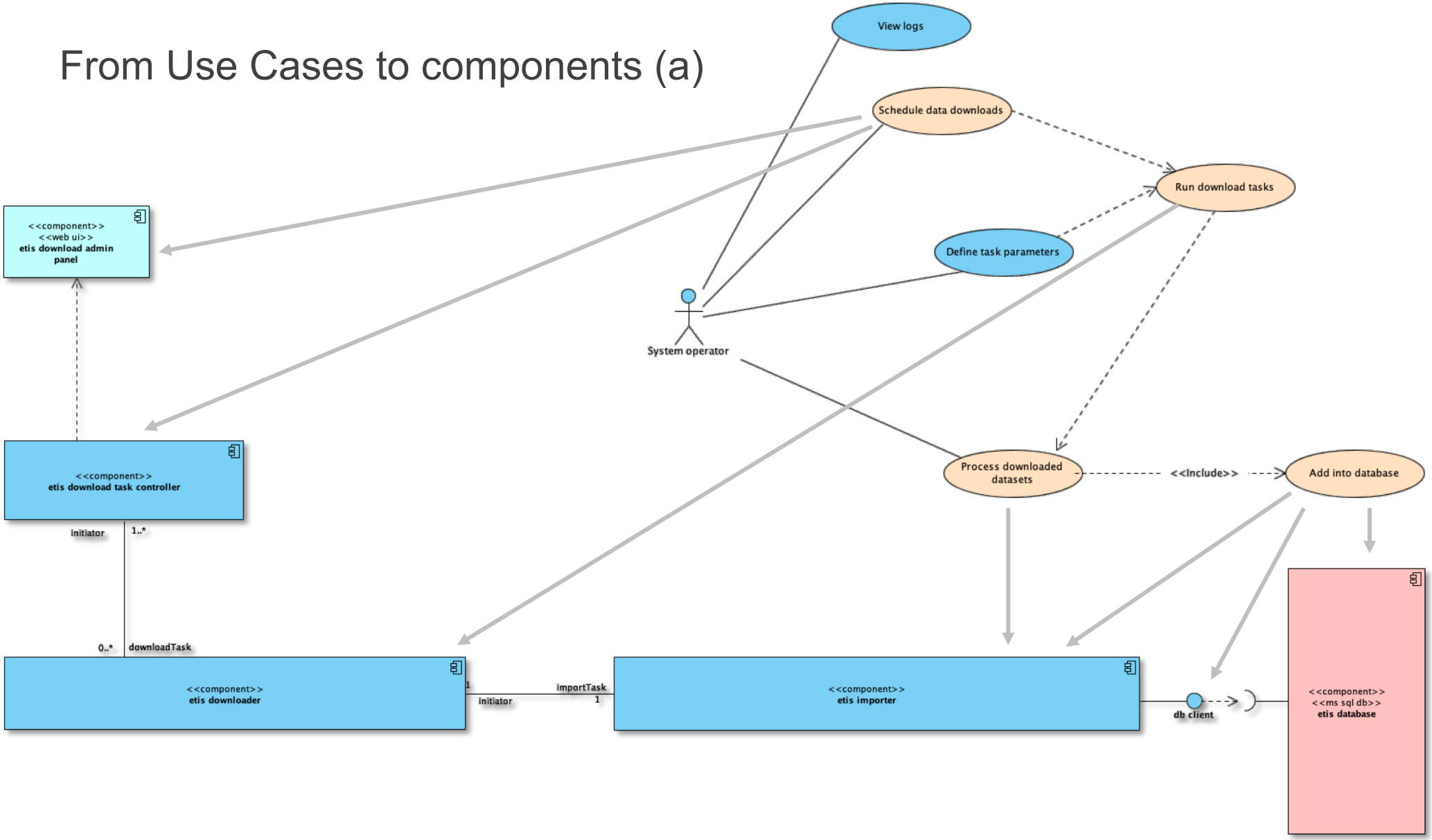
A real-life example: energy information system

A conceptual (but useful!) deployment diagram



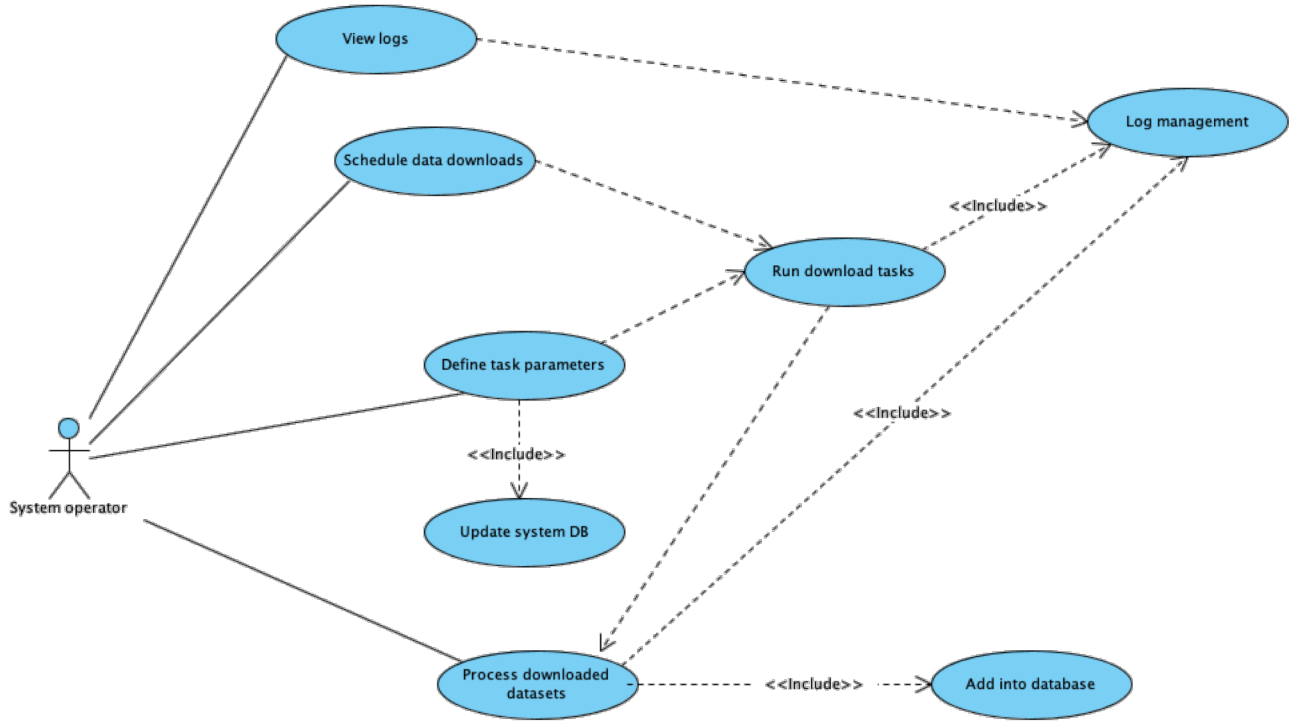
A real-life example: energy information system

From Use Cases to components (a)



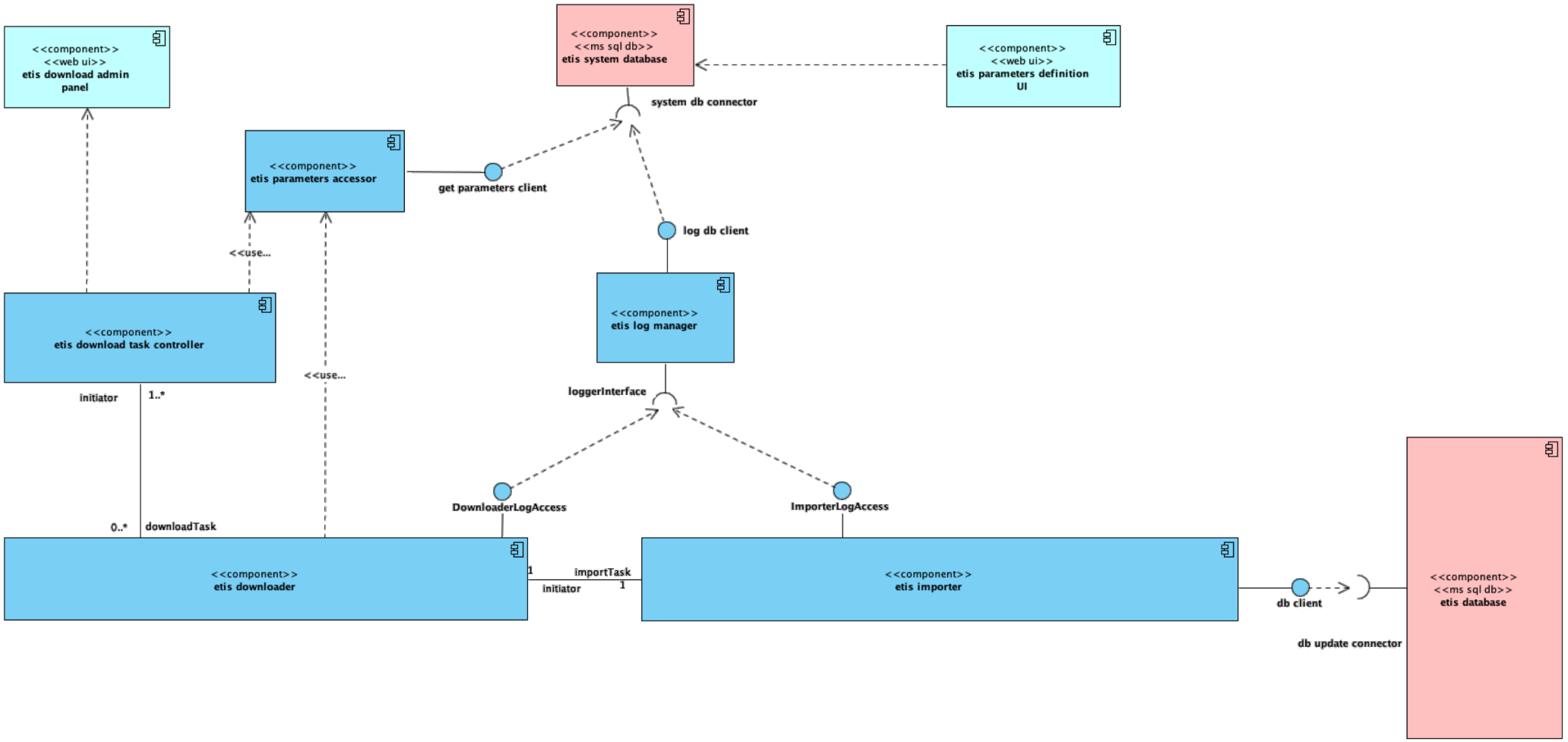
A real-life example: energy information system

From Use Cases to components (b)

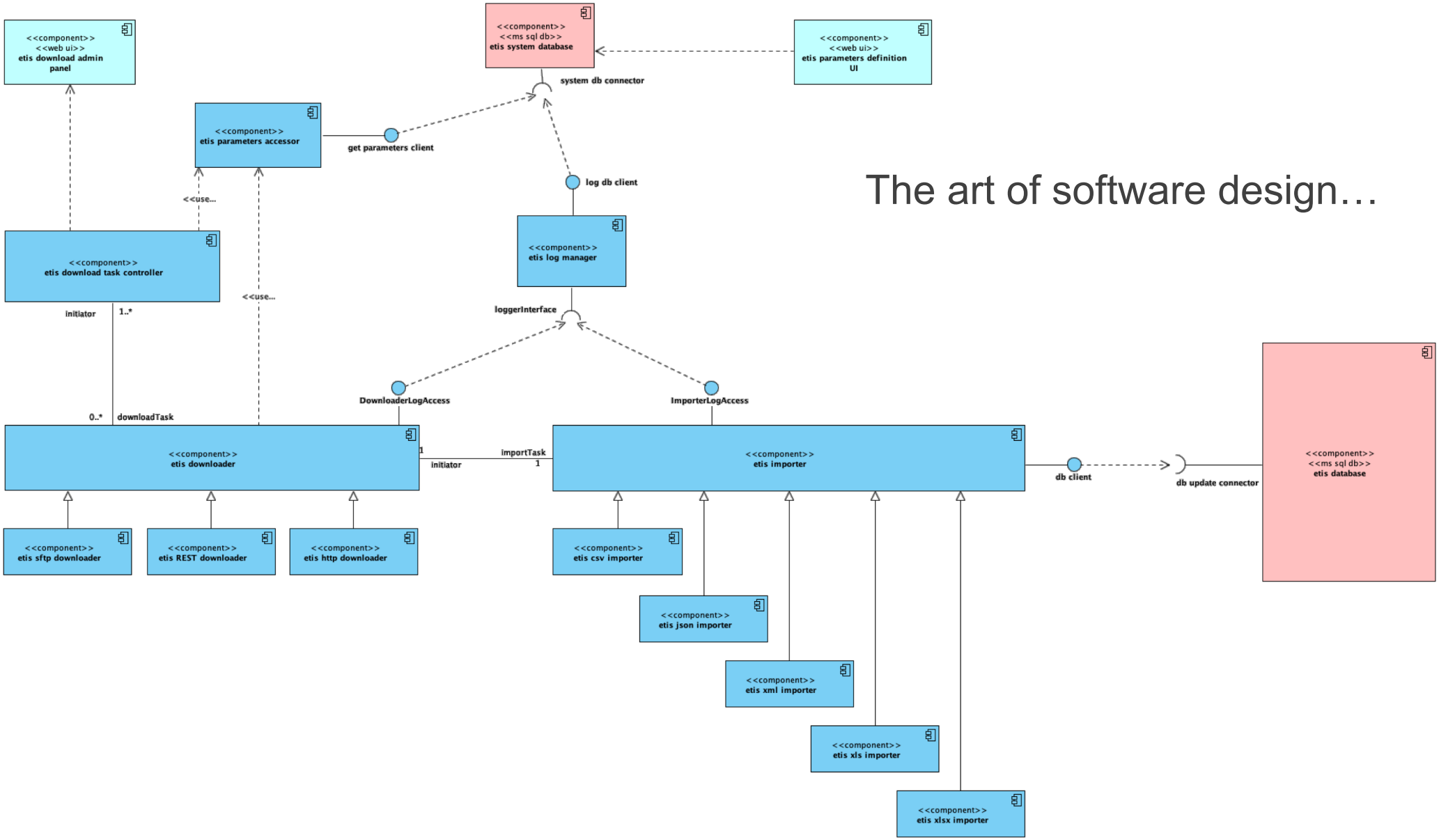


A real-life example: energy information system

From Use Cases to components (b)



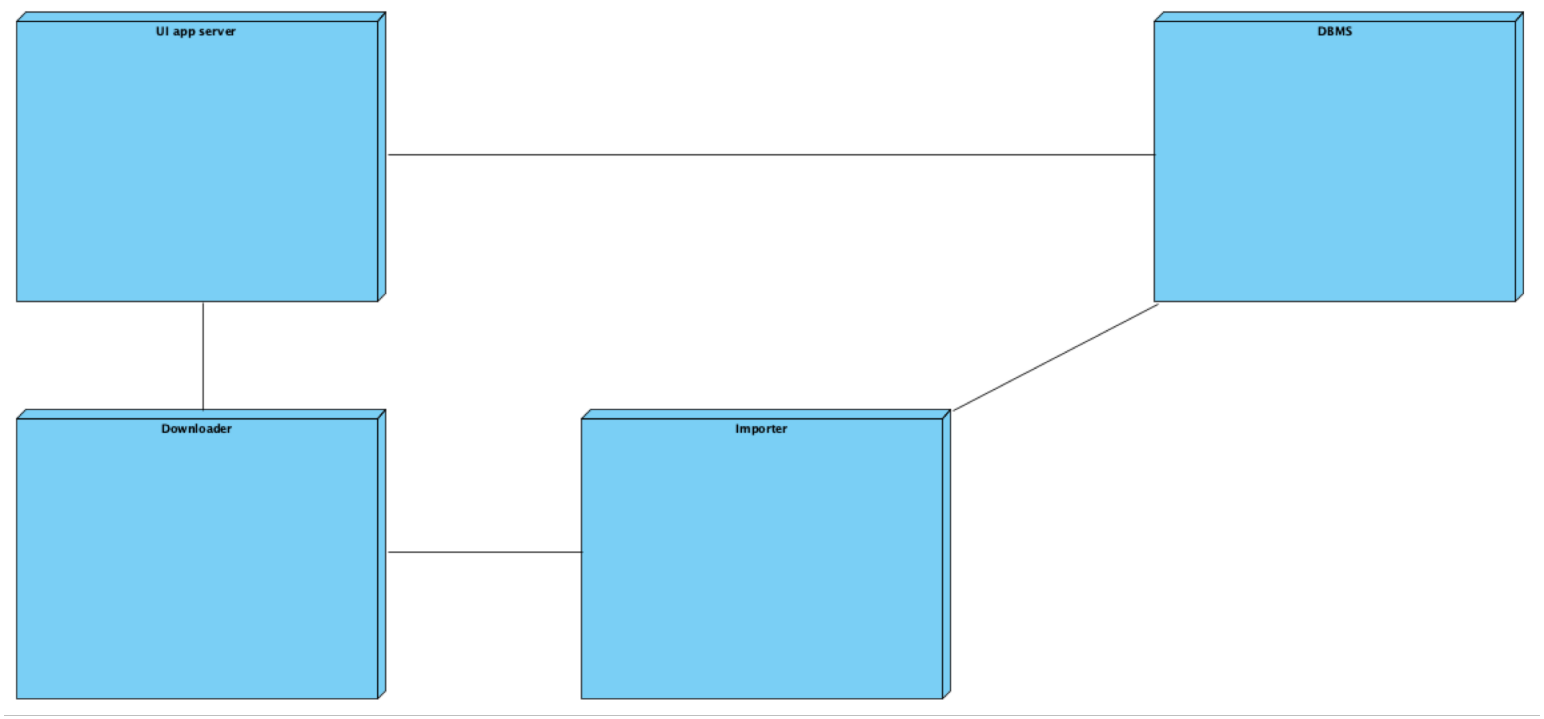
A real-life example: energy information system



The art of software design…

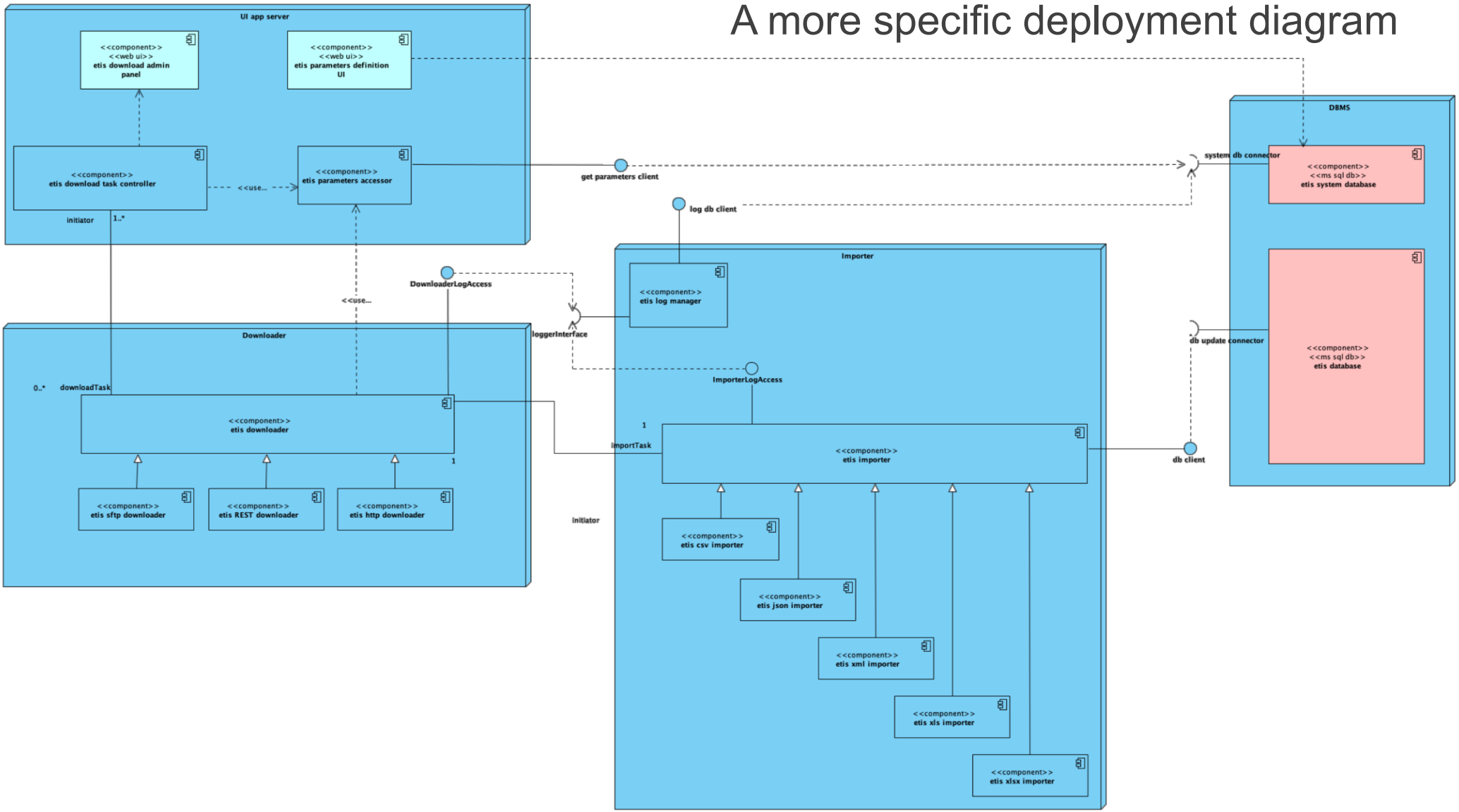
A real-life example: energy information system

Let’s make the deployment diagram more specific

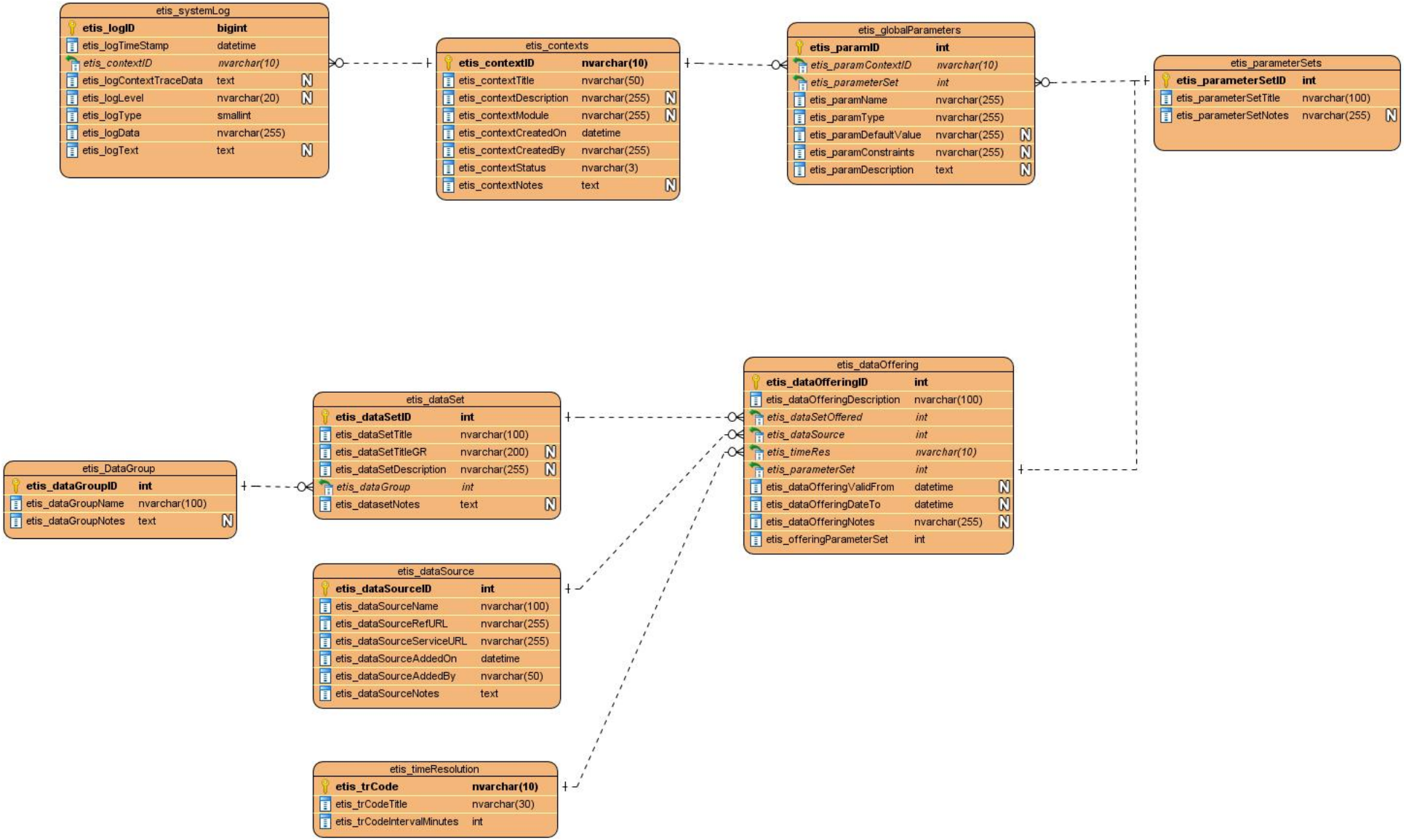


A real-life example: energy information system

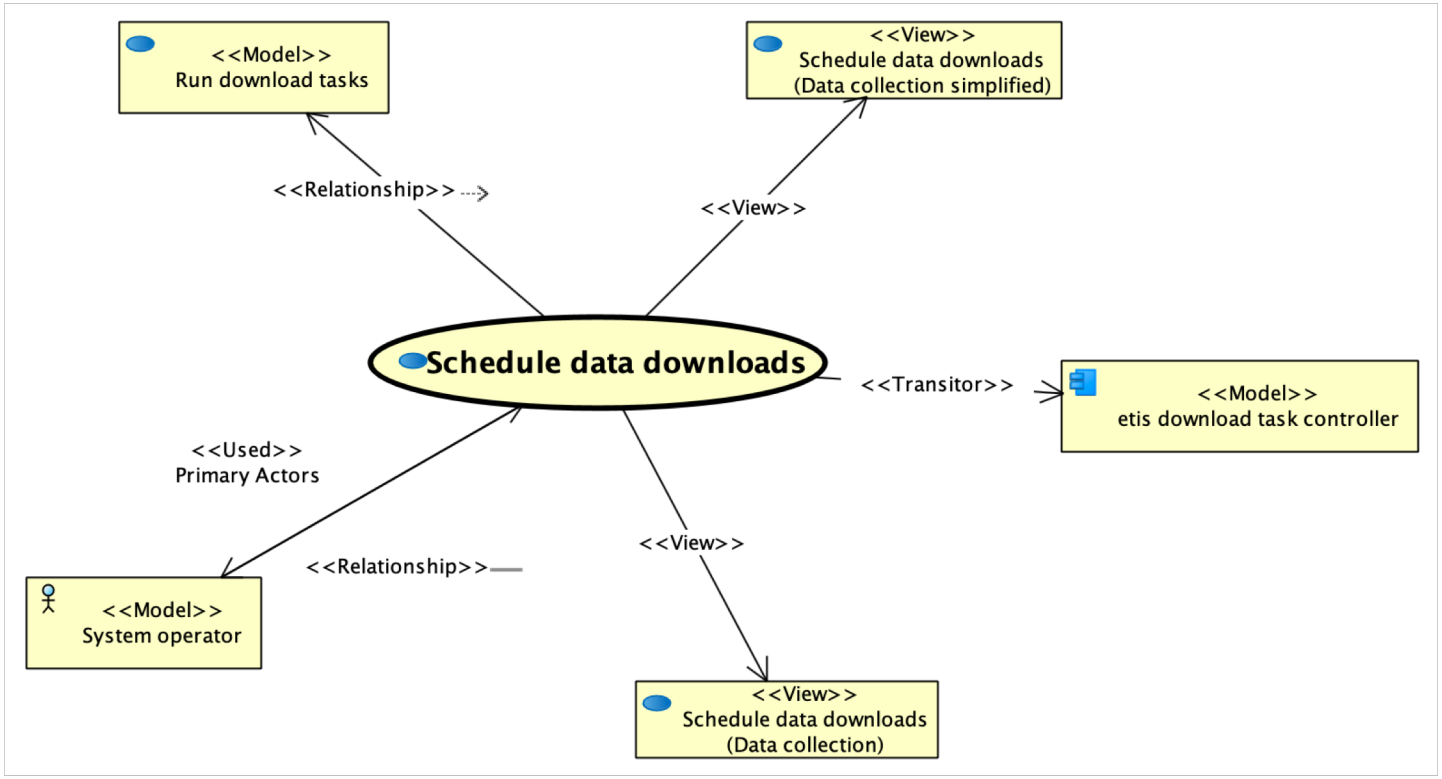
A more specific deployment diagram



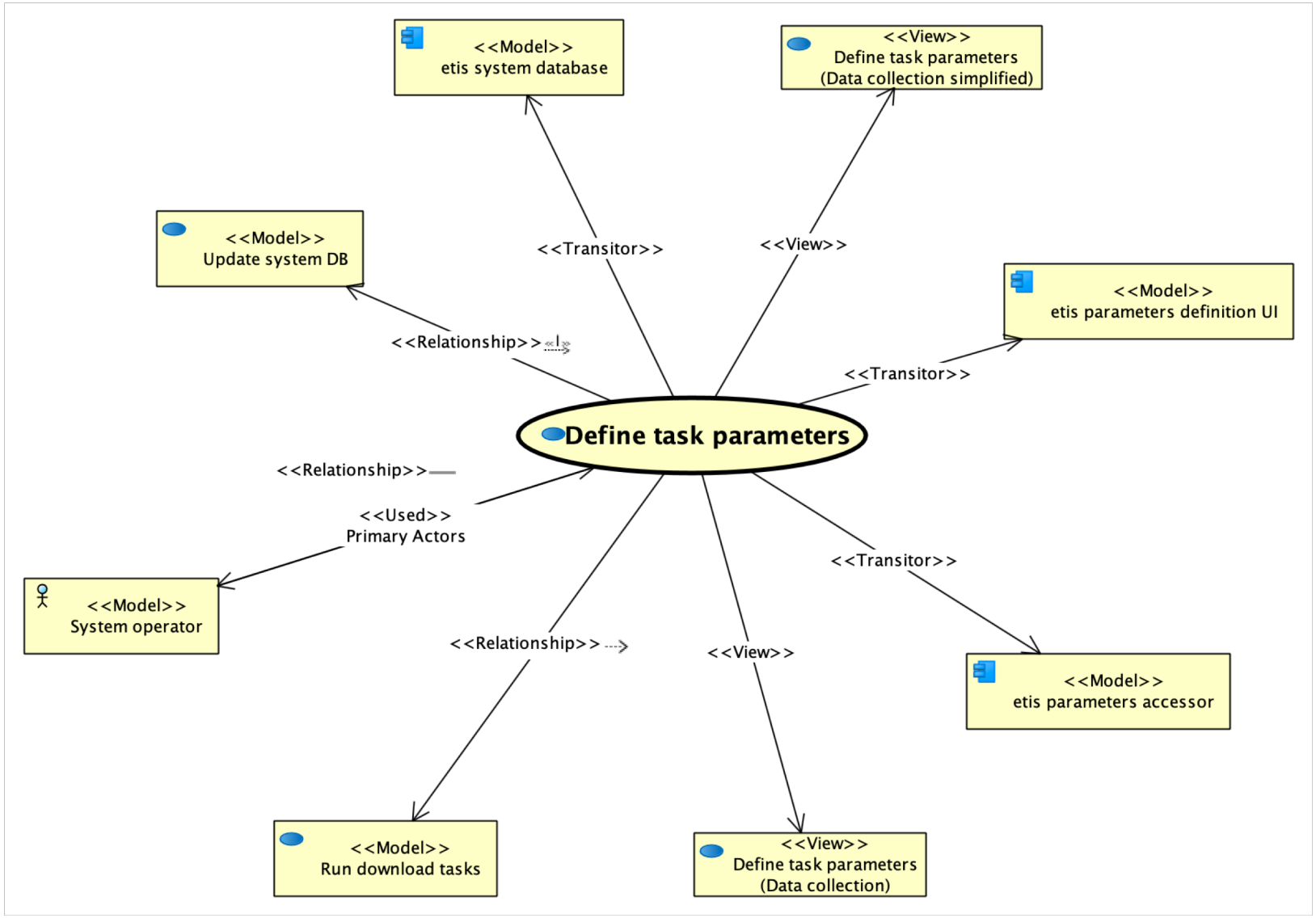
A real-life example: energy information system

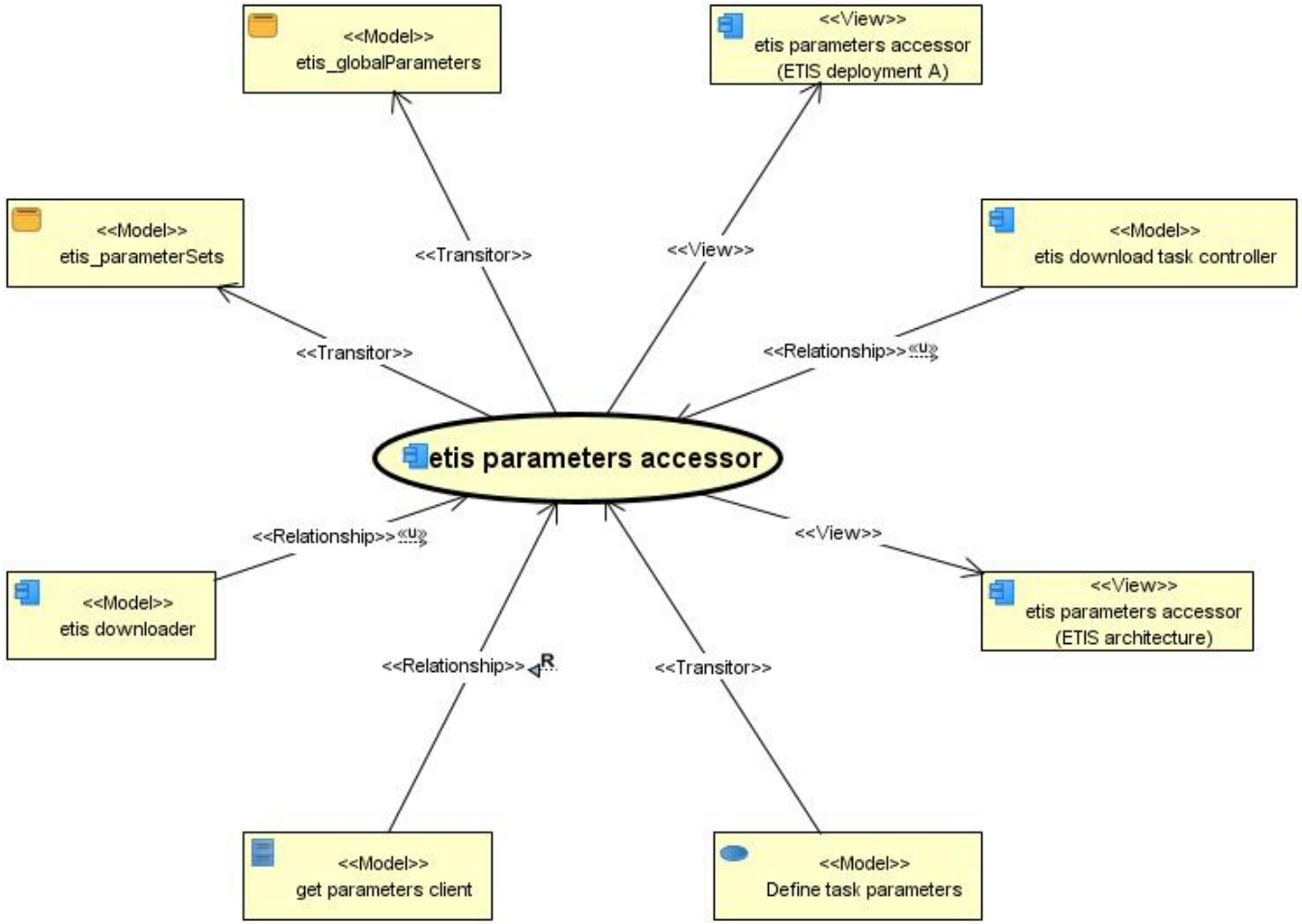


A UML model is more than the sum of its diagrams



A UML model is more than the sum of its diagrams




<<Model>>
etis_globalParameters
<<View>>
etis parameters accessor
(ETIS deployment A)
<<Model>>
etis_parameterSets
<<Transitor>>
<<View>>
<<Model>>
etis download task controller
<<Transitor>>
<<Relationship>>«u»
etis parameters accessor
<<Relationship>>«u»
<<View>>
<<Model>>
etis downloader
<<View>>
etis parameters accessor
(ETIS architecture)
<<Relationship>>R
<<Transitor>>
<<Model>>
get parameters client
<<Model>>
Define task parameters