Project Funding

Software Engineering Project SS2020

Name: Md. Shamsul Haque

Matriculation Number: 3070938

Group: 16

Lab Time: Fri. 10-12

Date of submission: 15.09.2020

Contents

1 Analysis

1 Analysis

1.1 A1

1.1.1 Requirements and Domain Konwledge (Assumption and Facts):

Requirements:

R1: A project starter can enter a funding request for a project on the platform.

R2: The project starter must provide his/her email address and payment information, a project name and description for a funding request.

R3: Project starters can cancel their own open projects any time.

R4: Supporters can search for open, successful, and failed projects and view their details.

R5: A supporter must enter his/her email address and payment information, and the amount of money he/she wants to donate for the donation.

R6: A supporter can cancel the donation till the project is open.

R7: If a supporter likes a project, then he/she shall be able to donate for the project.

R8: After the funding request the confirmation link must be sent to the users in provided email.

R9: By clicking the links that provided in users email, user can confirm or cancel his/her request or donation.

R10: If the end date of a project is reached, then the software shall mark the project either as successful if the funding limit was reached or as failed otherwise. In the case that a project failed, then all supporters and the project starter are informed. In the case that a project is successful, then also all supporters and the project starter are informed and additionally the supporters are charged using their payment information and the donated money is transferred to the project starter using his/her payment information.

R11: The machine must to generate random links every time.

R12: There must be a fixed time duration, so that the project starter can request the project.

R13: After the deadline supporters cannot be able to donate or cancel the donation request.

Assumption:

A1: After submitting proper information of the project, the project starter has higher chance to get the donation.

A2: A project starter can enter a funding request, but it is not guaranteed that he/ she will get the donation from supporters.

A3: After proving the valid email and payment information, project starter will be ready for funding request.

A4: Project starter and Supporter are able to use a web browser and have access to the internet.

A5: Project starter and Supporter normally confirm or change their decisions by using the links which have been sent to their email.

A6: By randomly generated link, other users are unable to identify or misuse the link.

A7: Users regularly checks, whether they get the confirmation links.

A8: Users need not to register to use the platform.

Facts:

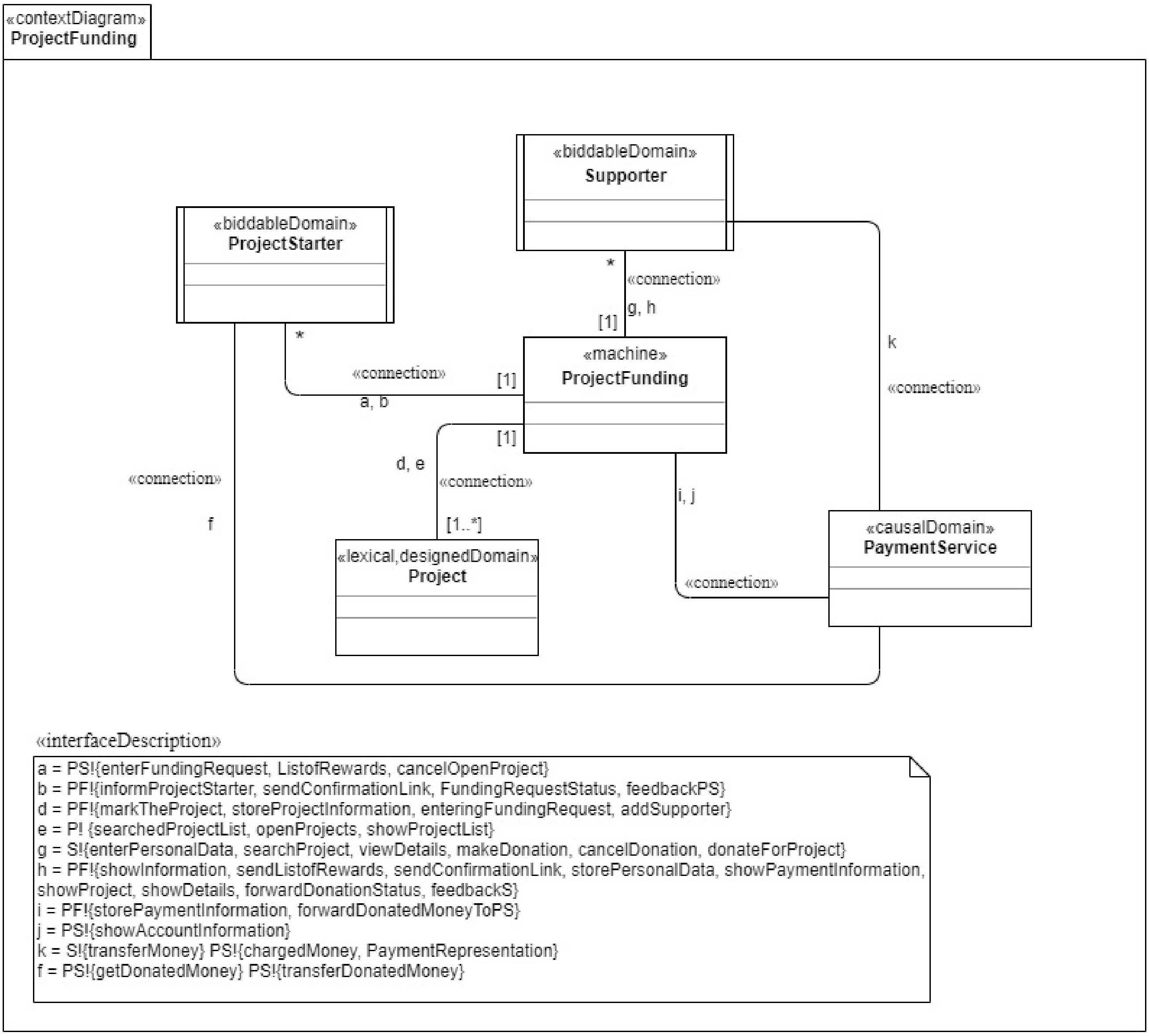
F1: A project is called open, when its funding limit and end date is not yet reached.

F2: It is possible to access a web page by using an internet connection and a web browser.

F3: A funding limit is needed to start a project.

F4: There are two types of users: Project starter and Supporter.

1.1.2 Context Diagram:



«interfaceDescription»

a = PS!{enterFundingRequest, ListofRewards, cancelOpenProject}
b = PF!{informProjectStarter, sendConfirmationLink, FundingRequestStatus, feedbackPS}
d = PF!{markTheProject, storeProjectInformation, enteringFundingRequest, addSupporter}
e = P! {searchedProjectList, openProjects, showProjectList}
g = S!{enterPersonalData, searchProject, viewDetails, makeDonation, cancelDonation, donateForProject}
h = PF!{showInformation, sendListofRewards, sendConfirmationLink, storePersonalData, showPaymentInformation, showProject, showDetails, forwardDonationStatus, feedbackS}
i = PF!{storePaymentInformation, forwardDonatedMoneyToPS}
j = PS!{showAccountInformation}
k = S!{transferMoney} PS!{chargedMoney, PaymentRepresentation}
f = PS!{getDonatedMoney} PS!{transferDonatedMoney}

Figure 1.1: Context Diagram

1.1.3 Validation:

- The glossary contains the notions used in R and D. The notions mentioned in R and D are contained in the glossary.
- Domains and phenomena of the context diagram must be consistent with R and D.

Notation in CD	Notation in R/D	Type
enter Funding request	enter a funding request for project	phenomenon
inform users	inform users for project's success or failure	phenomenon
send confirmation link	the confirmation link must be sent to users	phenomenon
cancel open project	can cancel their own open project	phenomenon
Project Starter	User who will do the project	Domain
Project Funding	The software we use for funding	Domain
mark the project	Software must mark the project	phenomenon
search for project	Supporters can search for open projects	phenomenon
supporters are charged	supporters will be charged by their payment information	phenomenon
Supporter	User who will donate for the project	Domain
Project	Project	Domain
transfer money	transfer money to Project Starter	phenomenon
cancel Donation	cancel donate request	phenomenon
Payment Service	existing payment service	Domain
enter payment information	provide payment information	phenomenon
Donation	donated money	Domain
list of rewards	provide list of rewards	phenomenon

- There must be exactly one context diagram Only one context diagram is provided.
- A context diagram has at least one machine domain. ProjectFunding is one machine domain.

Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
ProjectFunding	machine domain	ProjectStarter	biddable domain
		Supporter	biddable domain
		Project	lexical domain, designed domain
		PaymentService	causal domain
ProjectStarter	biddable domain	ProjectFunding	machine domain
		PaymentService	causal domain
Supporter	biddable domain	ProjectFunding	machine domain
		PaymentService	causal domain
Project	lexical domain, designed domain	ProjectFunding	machine domain
PaymentService	causal domain	ProjectFunding	machine domain
		ProjectStarter	biddable domain
		Supporter	biddable domain

- The machine domain must control at least one interface. ProjectFunding controls several interfaces (feedbackPS, feedbackS, ...)
- Biddable domains cannot be directly connected to lexical domains. No biddable domain is connected to a lexical domain.

Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
ProjectFunding	machine domain	ProjectStarter	biddable domain
		Supporter	biddable domain
		Project	lexical domain, designed domain
		PaymentService	causal domain
ProjectStarter	biddable domain	ProjectFunding	machine domain
		PaymentService	causal domain
Supporter	biddable domain	ProjectFunding	machine domain
		PaymentService	causal domain
Project	lexical domain, designed domain	ProjectFunding	machine domain
PaymentService	causal domain	ProjectFunding	machine domain
		ProjectStarter	biddable domain
		Supporter	biddable domain

- Causal, designed, lexical, display, machine domain type are not allowed together with biddable domain. ProjectStarter and Supporter are biddable domains only.

• Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
ProjectFunding	machine domain	ProjectStarter	biddable domain
		Supporter	biddable domain
		Project	lexical domain, designed domain
		PaymentService	causal domain
ProjectStarter	biddable domain	ProjectFunding	machine domain
		PaymentService	causal domain
Supporter	biddable domain	ProjectFunding	machine domain
		PaymentService	causal domain
Project	lexical domain, designed domain	ProjectFunding	machine domain
PaymentService	causal domain	ProjectFunding	machine domain
		ProjectStarter	biddable domain
		Supporter	biddable domain

- Phenomena controlled by a biddable domain must have counterpart phenomena located between machine and causal/lexical/designed domains.

	biddable domain phenomena	counterpart
ProjectStarter	enterFundingRequest	enteringFundingRequest
	ListofRewards	sendListofRewards
	cancelOpenProject	informUser
Supporter	enterPersonalData	storePersonalData
	searchProject, viewDetails	showProject, showDetails
	make Donation, cancelDonation	storeProjectInformation
	transferMoney	showPaymentInformation

- Connection and display domains must have at least one observed and one controlled interface.
- For each phenomenon controlled by a connection domain, there must be at least one phenomenon controlled by one of the connected domains.
- For each phenomenon observed by a connection or display domain, there must be at least one phenomenon controlled the connection or display domain.
- Context diagram contains no connection domain and no display domain.

1.2 A2

1.2.1 Problem Diagram with mapping.

R01: A project starter can enter a funding request for a project on the platform.

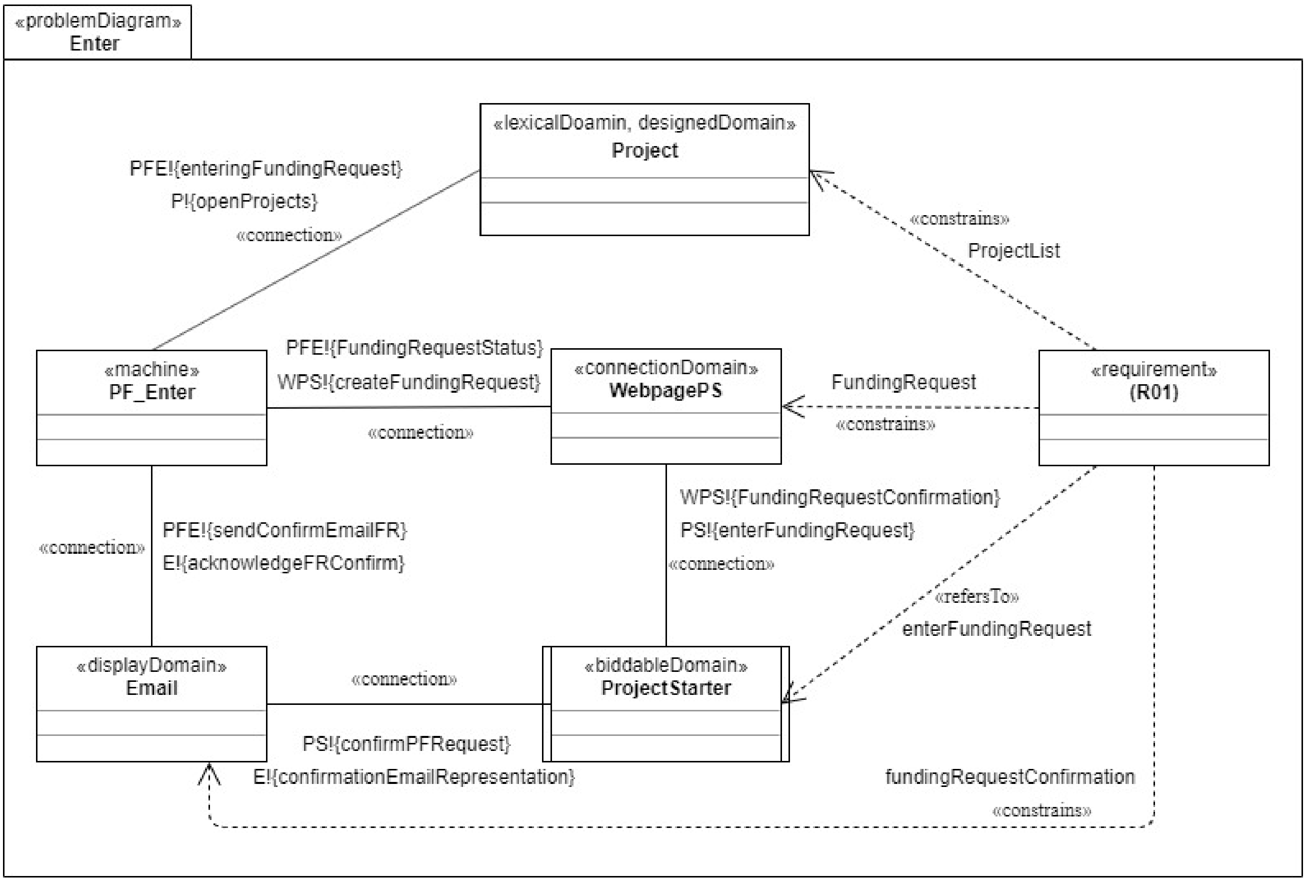


Figure 1.2.1: Problem Diagram for R01

Description of Problem Diagram for R04:

The above diagram shows for the problem of entering funding request for a project. We make this diagram so that a projectStarter can enter his funding request in the platform. Here Email display Domain is shown so that confirmation can be sent and displayed to user. Here we use connection Domain because our biddable Domain can interact with machine.

Concretize interface(s):

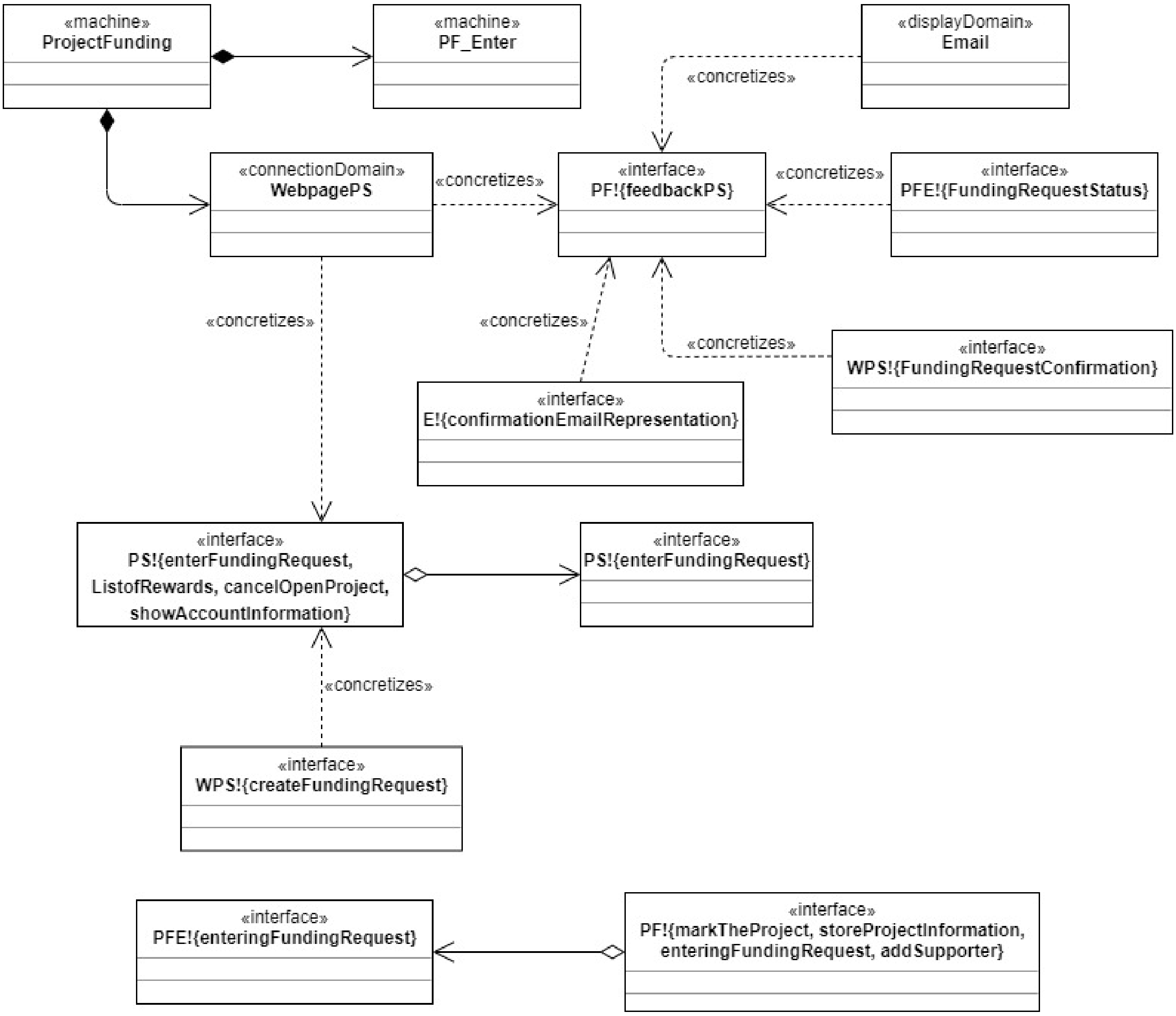


Figure: Problem Diagram's mapping for R01

Above figure is the mapping for R01. Here We concretize PF!{…} and PS!{…} domain. We splited up here PF!{…} and PS!{…}.

R04: Supporters can search for open, successful, and failed projects and view their details.

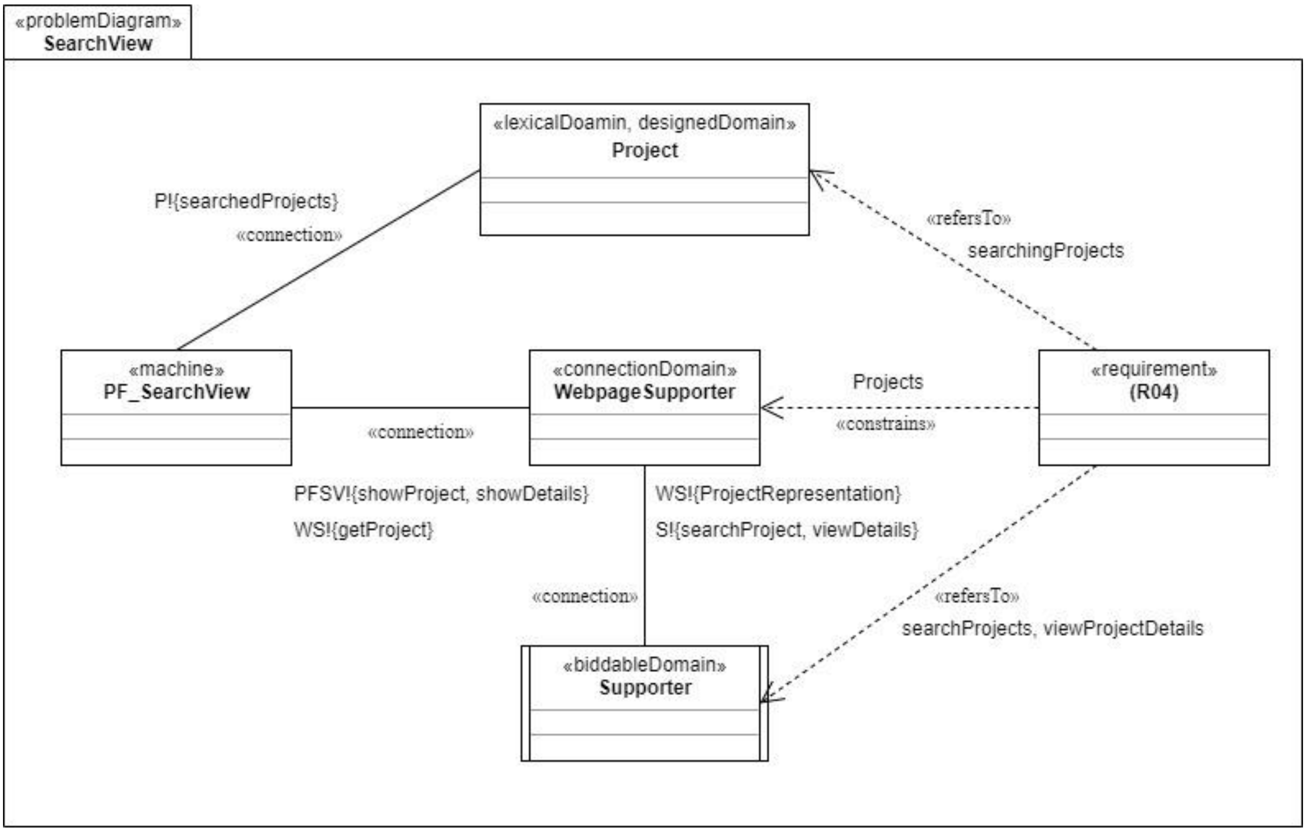


Figure 1.2.2: Problem Diagram for R04

Description of Problem Diagram for R04:

The above diagram shows for the problem of searching open, failed and successful projects. We make this diagram so that a supporter can search for his/her desired project and can view their details. Here we use connection Domain because our biddable Domain can interact with machine.

Concretize interface(s):

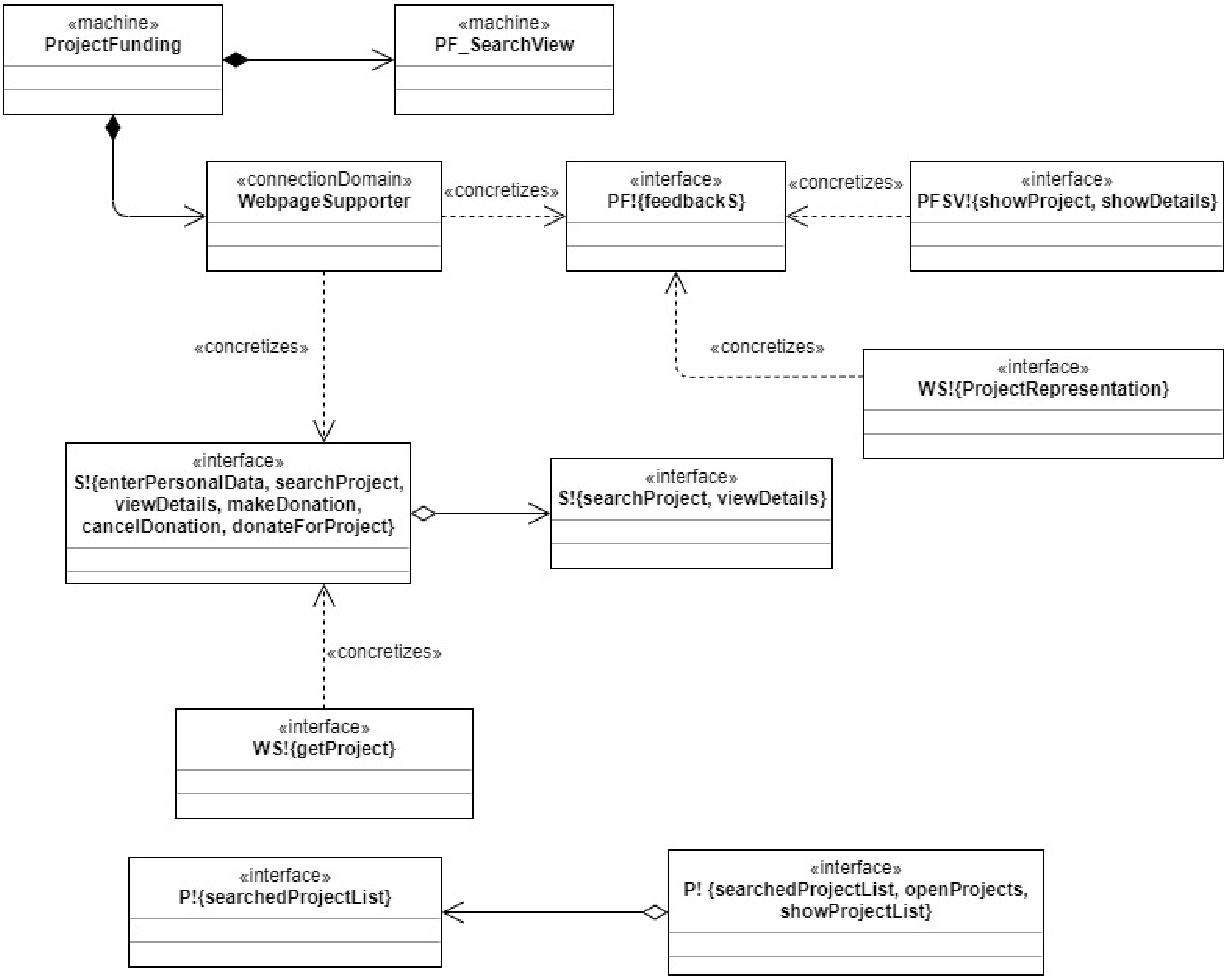
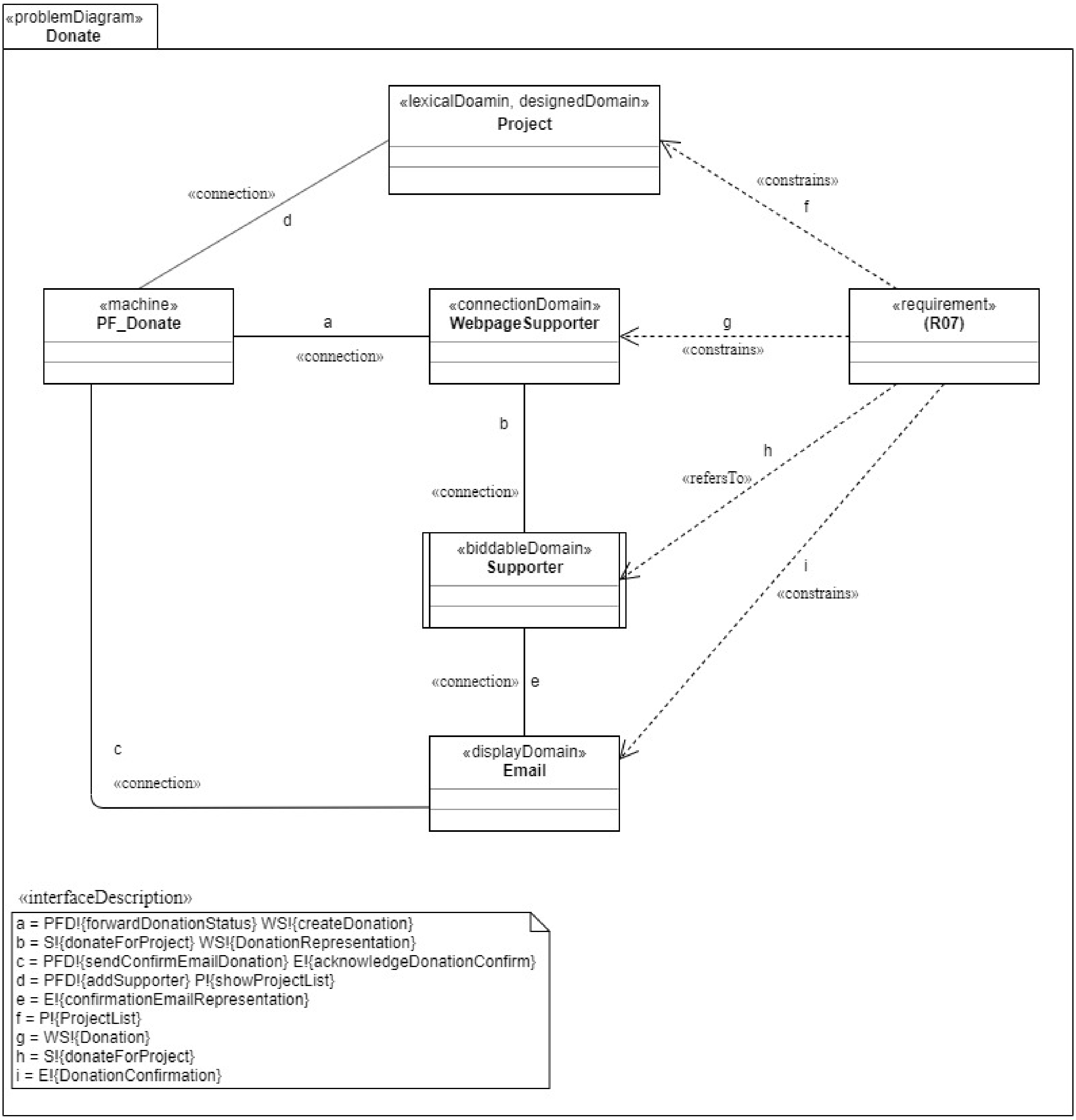


Figure: Problem Diagram's mapping for R04

Above figure is the mapping for R04. Here We concretize PF!{…} and S!{…} domain. We splited up here P!{…} and S!{…}.

R07: If a supporter likes a project, then he/she shall be able to donate for the project.



a = PFD!{forwardDonationStatus} WS!{createDonation}
b = S!{donateForProject} WS!{DonationRepresentation}
c = PFD!{sendConfirmEmailDonation} E!{acknowledgeDonationConfirm}
d = PFD!{addSupporter} P!{showProjectList}
e = E!{confirmationEmailRepresentation}
f = P!{ProjectList}
g = WS!{Donation}
h = S!{donateForProject}
i = E!{DonationConfirmation}

Figure 1.2.3: Problem Diagram for R07

Description of Problem Diagram for R07:

The above diagram shows for the problem of liking a project and can choose a project where the supporter feels interested. We make this diagram so that a supporter choose a project on which he/she might donate. Here Email display Domain is shown so that confirmation can be sent and displayed to user. Here we use connection Domain because our biddable Domain can interact with machine.

Concretize interface(s):

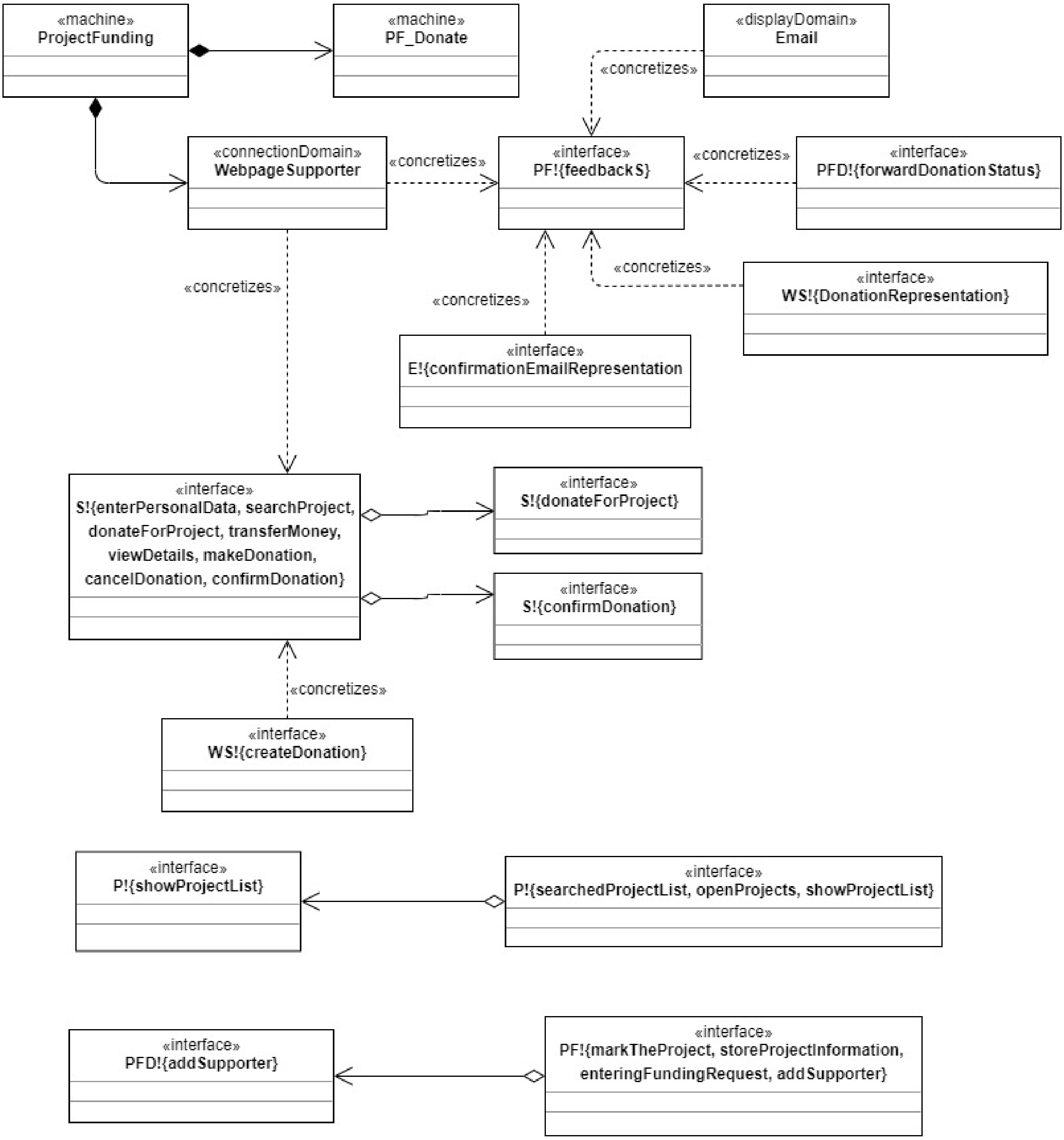
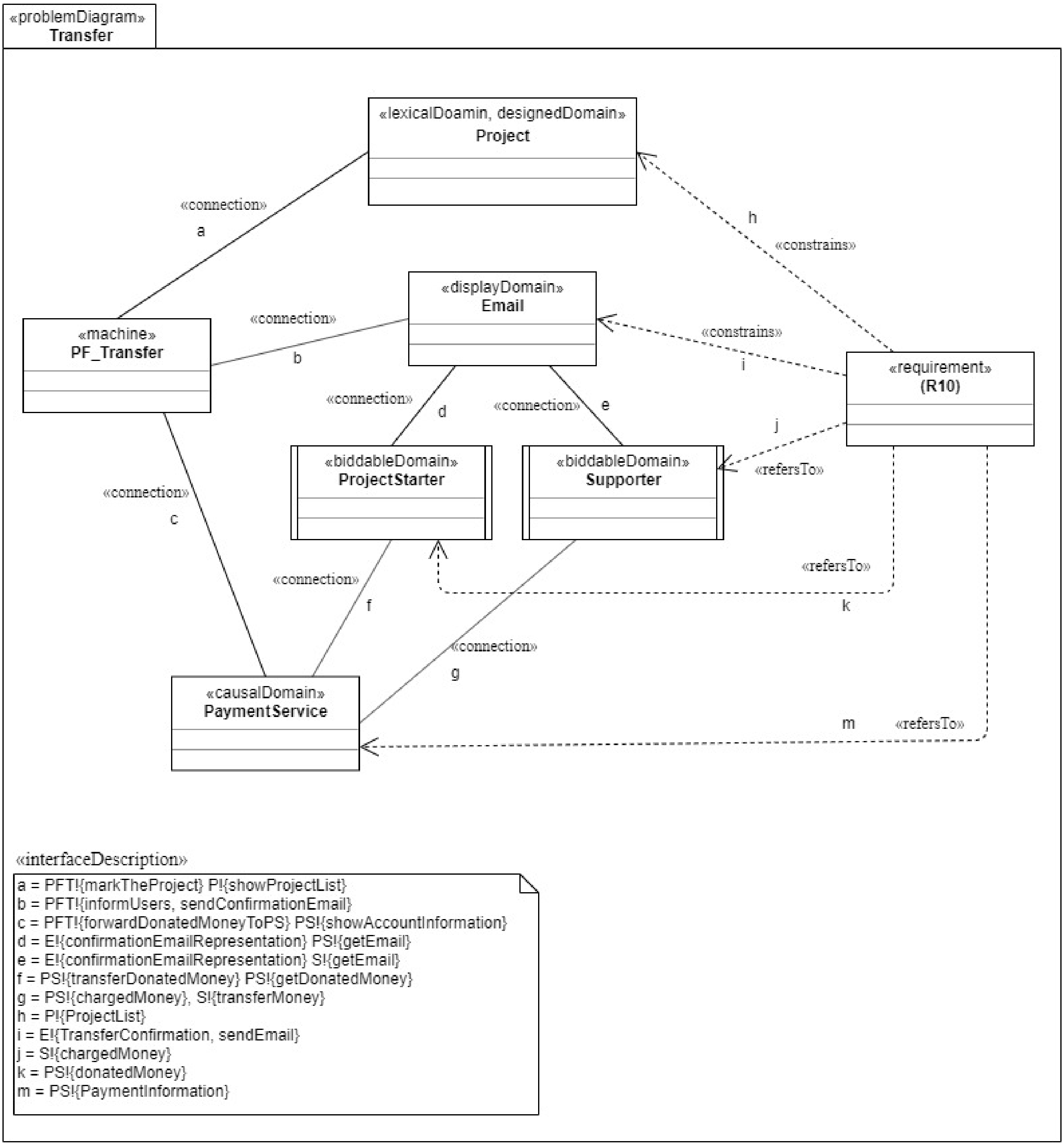


Figure: Problem Diagram’s mapping for R07

Above figure is the mapping for R07. Here We concretize PF!{…} and S!{…} domain. We splited up here P!{…}, S!{…} and PF!{…}.

R10: If the end date of a project is reached, then the software shall mark the project either as successful if the funding limit was reached or as failed otherwise. In the case that a project failed, then all supporters and the project starter are informed. In the case that a project is successful, then also all supporters and the project starter are informed and additionally the supporters are charged using their payment information and the donated money is transferred to the project starter using his/her payment information.



a = PFT!{markTheProject} P!{showProjectList}
b = PFT!{informUsers, sendConfirmationEmail}
c = PFT!{forwardDonatedMoneyToPS} PS!{showAccountInformation}
d = E!{confirmationEmailRepresentation} PS!{getEmail}
e = E!{confirmationEmailRepresentation} S!{getEmail}
f = PS!{transferDonatedMoney} PS!{getDonatedMoney}
g = PS!{chargedMoney}, S!{transferMoney}
h = P!{ProjectList}
i = E!{TransferConfirmation, sendEmail}
j = S!{chargedMoney}
k = PS!{donatedMoney}
m = PS!{PaymentInformation}

Figure 1.2.4: Problem Diagram for R10

Description of Problem Diagram for R10:

The above diagram shows for the problem of marking the project either successful or failed. Moreover the problem for making the scope of donation. We make this diagram so that a supporter can donate for a project via a paymentService where the payment informations are already stored. Then the donated money will be transferred to ProjectStarter account. Here Email display Domain is shown so that confirmation can be sent and displayed to user. Email is also used to inform the user in case of successful or failure of the project. Here we use no connection Domain because our biddable Domain are not directly interacting with machine.

Concretize interface(s):

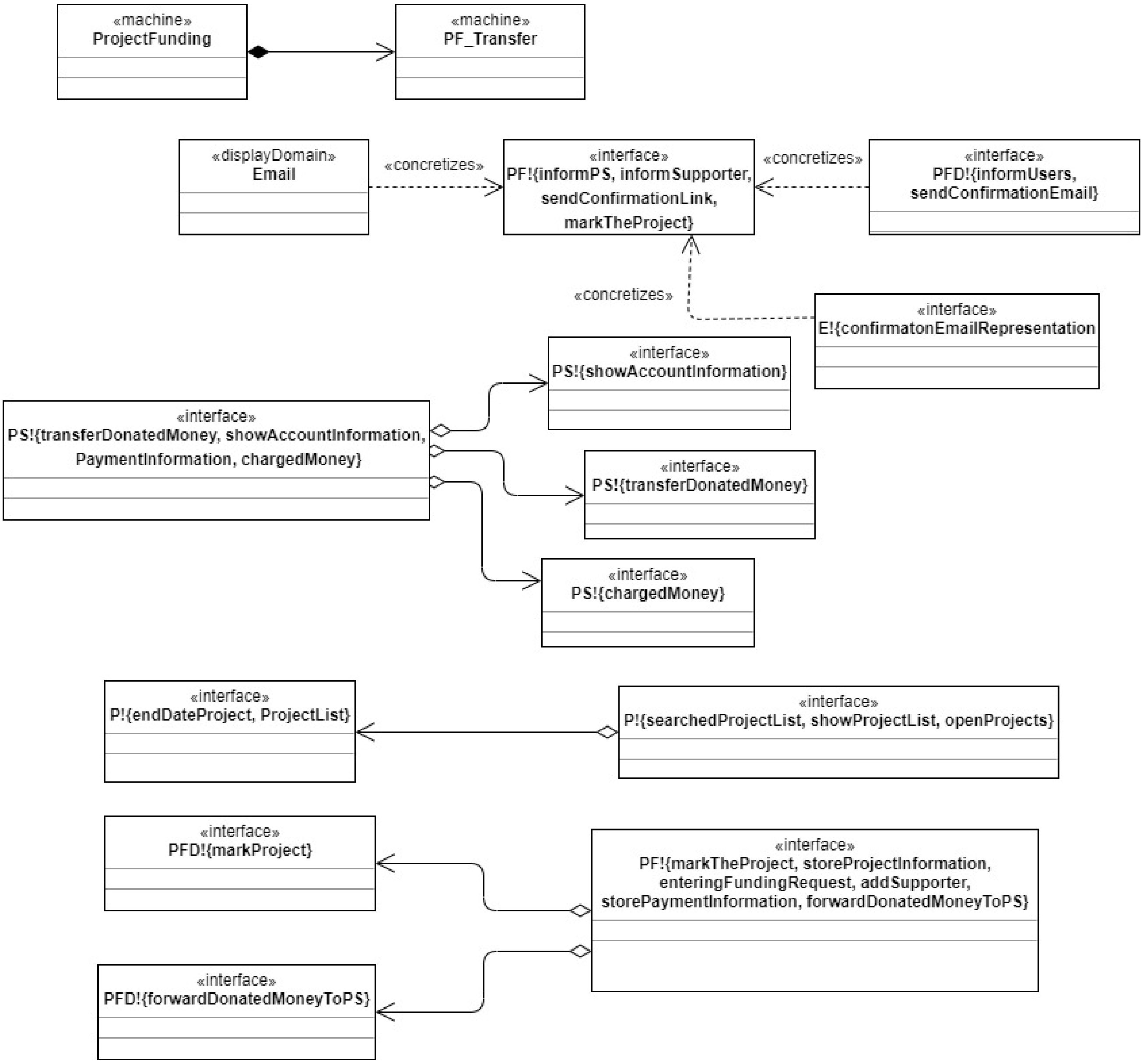

«machine»
ProjectFunding
«machine»
PF_Transfer
«displayDomain»
Email
«concretizes»
«interface»
PF!{informPS, informSupporter, sendConfirmationLink, markTheProject}
«concretizes»
«interface»
PFD!{informUsers, sendConfirmationEmail}
«concretizes»
«interface»
E!{confirmatonEmailRepresentation
«interface»
PS!{showAccountInformation}
«interface»
PS!{transferDonatedMoney, showAccountInformation, PaymentInformation, chargedMoney}
«interface»
PS!{transferDonatedMoney}
«interface»
PS!{chargedMoney}
«interface»
P!{endDateProject, ProjectList}
«interface»
P!{searchedProjectList, showProjectList, openProjects}
«interface»
PFD!{markProject}
«interface»
PF!{markTheProject, storeProjectInformation, enteringFundingRequest, addSupporter, storePaymentInformation, forwardDonatedMoneyToPS}
«interface»
PFD!{forwardDonatedMoneyToPS}

Figure: Problem Diagram's mapping for R10

Above figure is the mapping for R10. Here We concretize PF!{…} domain. We splited up here P!{…}, PF!{…} ans PS!{…}.

1.2.2 Problem Frames

Legend for the following table:

X: lexical domain, B: biddable domain, C: causal domain, D: display domain, CON: connection domain

Requirements	Name	constrained domain	referred to domain
R01	update 2	X, CON	B
R04	query 2	CON	B, X
R07	update 2	X, CON	B
R10	simple transformation	X	-

R01 can also be update 1 as there is display Domain and it is constrained by machine.

R07 can also be update 1 as there is display Domain and it is constrained by machine.

According to the table from script, R10 can also be information display as D is constrained and C refersTo.

1.2.3 Validation

- All requirements R are covered in some subproblem.

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest, FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails
		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation

		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList
		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation
		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney
		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- A problem diagram has exactly one machine domain.

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest, FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails
		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation
		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList
		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation

		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney
		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- A problem diagram contains at least one requirement.

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest, FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails
		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation
		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList
		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation
		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney

		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- The machine domain must control at least one interface.

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest, FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails
		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation
		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList
		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation
		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney
		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- Requirements constrain at least one domain.

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest,

					FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails
		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation
		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList
		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation
		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney
		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- Requirements do not constrain machine(s).

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest, FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails

		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation
		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList
		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation
		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney
		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- If requirements do constrain biddable domains, a good argument is given and documented.

Requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdEnter	PF_Enter	machine		enteringFundingRequest, FundingRequestStatus
		Project	lexical, designed	X	openProjects
		WebpagePS	connection	X	createFundingRequest, FundingRequestConfirmation
		Email	display	X	acknowledgeFRConfirm, confirmationEmailRepresentation
		ProjectStarter	biddable		enterFundingRequest
R04	pdSearchView	PF_SearchView	machine		showProject, showDetails
		Project	lexical, designed		searchedProjects
		WebpageSupporter	connection	X	getProject, ProjectRepresentation
		Supporter	biddable		searchProject, viewDetails
R07	pdDonate	PF_Donate	machine		forwardDonationStatus, sendConfirmEmailDonation, addSupporter
		Project	lexical, designed	X	showProjectList

		WebpageSupporter	connection	X	createDonation, DonationRepresentation
		Supporter	biddable		donateForProject, confirmDonation
		Email	display	X	acknowledgeDonationConfirm, confirmationEmailRepresentation
R10	pdTransfer	PF_Transfer	machine		markTheProject, informUsers, sendConfirmationEmail, forwardDonatedMoneyToPS
		Project	lexical, designed	X	showProjectList
		Email	display	X	confirmationEmailRepresentation
		ProjectStarter	biddable		getEmail, getDonatedMoney
		Supporter	biddable		getEmail, transferMoney
		PaymentService	causal		showAccountInformation, transferDonatedMoney, chargedMoney

- Connection domains must have at least one observed and one controlled interface.

connection domain	phenomenon controlled by connection domain	connected domain	phenomenon controlled by connected domain
WebpagePS	createFundingRequest	PF_Enter	FundingRequestStatus
	FundingRequestConfirmation	ProjectStarter	enterFundingRequest
WebpageSupporter	getProject	PF_SearchView	showProject, showDetails
	ProjectRepresentation	Supporter	searchProject, viewDetails
	createDonation	PF_Donate	forwardDonationStatus
	DonationRepresentation	Supporter	donateForProject

- For each phenomenon controlled by a connection domain, there must be at least one phenomenon controlled by one of the connected domains.
- For each phenomenon observed by a connection domain, there must be at least one phenomenon controlled by the connection domain.

connection domain	phenomenon controlled by connection domain	connected domain	phenomenon controlled by connected domain
WebpagePS	createFundingRequest	PF_Enter	FundingRequestStatus
	FundingRequestConfirmation	ProjectStarter	enterFundingRequest
WebpageSupporter	getProject	PF_SearchView	showProject, showDetails
	ProjectRepresentation	Supporter	searchProject, viewDetails
	createDonation	PF_Donate	forwardDonationStatus
	DonationRepresentation	Supporter	donateForProject

- The problem diagrams must be consistent to the context diagram, e.g. each machine of the problem diagrams is a part of the context diagram machine.
  Provided mapping diagrams
- All subproblems can be derived from the context diagram by means of decomposition operators.

problem diagram	operator	related domains or phenomena
pdEnter	leave out domain	Supporter, PaymentService
	introduce connection/display domain	WebpagePS, Email
	split interface	PS!{…}, PF!{…}
	concretize interface	PS!{…}, PF!{…}
pdSearchView	leave out domain	ProjectStarter, PaymentService
	introduce connection/display domain	WebpageSupporter
	split interface	S!{…}, P!{…}
	concretize interface	S!{…}, PF!{…}
pdDonate	leave out domain	ProjectStarter, PaymentService
	introduce connection/display domain	WebpageSupporter, Email
	split interface	S!{…}, PF!{…}, P{…}
	concretize interface	S!{…}, PF!{…}
pdTransfer	leave out domain	
	introduce connection/display domain	Email
	split interface	PS!{…}, P!{…}, PF!{…}
	concretize interface	PS!{…}, PF!{…}

1.3 A3

1.3.1 Deriving the specifications.

R01: A project starter can enter a funding request for a project on the platform.

Using the domain knowledge:

Email (F05): Email transforms the command “sendConfirmationEmailFR” from the machine into the corresponding mail data “confirmationEmailRepresentation” for the Project starter. Then it transforms the command “confirmPFRequest” into the corresponding mail data “acknowledgeFRConfirm” for the machine.

we can derive the specifications:

Incoming Phenomena	**Caused Phenomena**
PS!{enterFundingRequest}	WPS!{createFundingRequest}
PFE!{FundingRequestStatus}	WPS!{FundingRequestConfirmation}
PFE!{enteringFundingRequest}	P!{openProjects}
PFE!{sendConfirmationEmailFR}	E!{confirmationEmailRepresentation}

WebpagePS (S01a): When the webpage receives the command “enterFundingRequest”, then the command is forwarded to the machine with the command “createFundingRequest”. The feedback whether the request was confirmed or not is received via the command “fundingRequestStatus”. This feedback is shown to the ProjectStarter via the command “fundingRequestConfirmation”.

PF_Enter(S01b): When the machine receives the command “createFundingRequest”, then it commands the Project for creating funding request with the command “ enteringFundingRequest ” and the result is received as the data “openProjects”. Then it sends the command “sendConfmEmailFR" to create the Confirmation email, and informs the Project Starter with the command “FundingRequestStatus” to the Project Starter`s web page about the Project creation process.

Project (S01c): After receiving the command “enteringFundingRequest” the results are returned as the data “openProjects”.

Correctness condition: (S01a) ^ (S01b) ^ (S01c) ^ (F05) =) (R01)

R04: Supporters can search for open, successful, and failed projects and view their details.

We can derive the following specications for R04:

Incoming Phenomena	**Caused Phenomena**
S!{searchProject, viewDetails}	WS!{getProject}
PFS!{showProject, showDetails}	WS!{ProjectRepresentation}

WebpageSupporter (S04a): When the webpage receives the command “searchprojects, viewDetails”, then the command is forwarded to the machine with the command “getProject”. The results are received via the command “showProject” and shown to the guest by “ProjectRepresentation”.

Pf_searchview (S04b) : When the machine receives the command “getProject”, the Projects are selected with the command “get_project” and received as the data “searchedProjects”. The results are returned via the command “showProject, show Details”.

Project (S04c) : After receiving the command “get_Project” the results are returned as the data “searchedProjects”.

Correctness Condition: (S04a) ∧ (S04b) ∧ (S04c) =⇒ (R04)

R07: If a supporter likes a project, then he/she shall be able to donate for the project.

We can derive the following specications for R07:
Using the domain knowledge

Email (F06): Email transforms the command “sendConfmEmailDonation"
from the machine into the corresponding mail data
“confirmationEmailRepresentation" for the Supporter.

we can derive the Specifications:

Incoming Phenomena	**Caused Phenomena**
S!{donateForProject}	WS!{createDonation}
PFD!{ forwardDonationStatus}	WS!{DonationRepresentation}
PFD!{addSupporter}	P!{showProjectList}
PFD!{sendConfirmationEmailDonation}	E!{confirmationEmailRepresentation}

WebpageSupporter (S07a): When the webpageSupporter receives the command "donateForProject", then the command is forwarded to the machine with the command "createDonation". The feedback whether the request was confirmed or not is received via the commands "forwardDonationStatus". Feedback is shown to the Supporter via the commands "DonationRepresentation".

PF_Donate(S07b): When the machine receives the command "createDonation", then it is checked if the Project is available with the command "get_showProjectList" and the result is received as the data "showProjectList". If the Project is available, then the machine sets it as Supporter with the command "addSupporter", sends the command "sendConfirmEmailDonation" to create the Confirmation email, and informs the Project Supporter with the command "forwardDonationStatus" to the Project Supporter`s web page about the Project donation process.

Project (S07c): After receiving the command "get_showprojectList" the results are returned as the data "showProjectList". When the command "addSupporter" is received, the project is marked as Liked for donation.

Correctness condition: (F06) ^ (S04a) ^ (S04b) ^ (S04c) =) (R04)

R10: If the end date of a project is reached, then the software shall mark the project either as successful if the funding limit was reached or as failed otherwise. In the case that a project failed, then all supporters and the project starter are informed. In the case that a project is successful, then also all supporters and the project starter are informed and additionally the supporters are charged using their payment information and the donated money is transferred to the project starter using his/her payment information.

Using the domain knowledge:

Email (F07): Email transforms the command "informUsers, sendConfirmationEmail" from the machine into the corresponding mail data "confirmationEmailRepresentation" for the both Project starter and Supporter.

we can derive the specifications:

Incoming Phenomena	**Caused Phenomena**
PFT!{informUsers, sendConfirmationEmail}	E!{confirmationEmailRepresentation}
PFT!{forwardDonatedMoneyToPS}	PS!{transferDonatedMoney}
PFT!{markTheProject}	P!{showProjectList}

PF_Transfer(S10a): When receiving the command "checkProject" all successful projects if the funding limit was reached after the end date is reached or failed projects if funding limit was not reached will be marked using the command "markProject".

Project(S10b): When receiving the command "markProject" all successful or failed Projects will be marked after the end date of project is reached.

PaymentService(S10c): When the PaymentService receives the command "forwardDonatedMoneyToPS", then it will transfer the charged money from Supporter to Project Starter.

Correctness condition: (S10a) ^ (S10b) ^ (S10c) ^ (F07) =) (R10)

1.3.2 Sequence Diagram for specification.

Sequence Diagram for specification (S01):

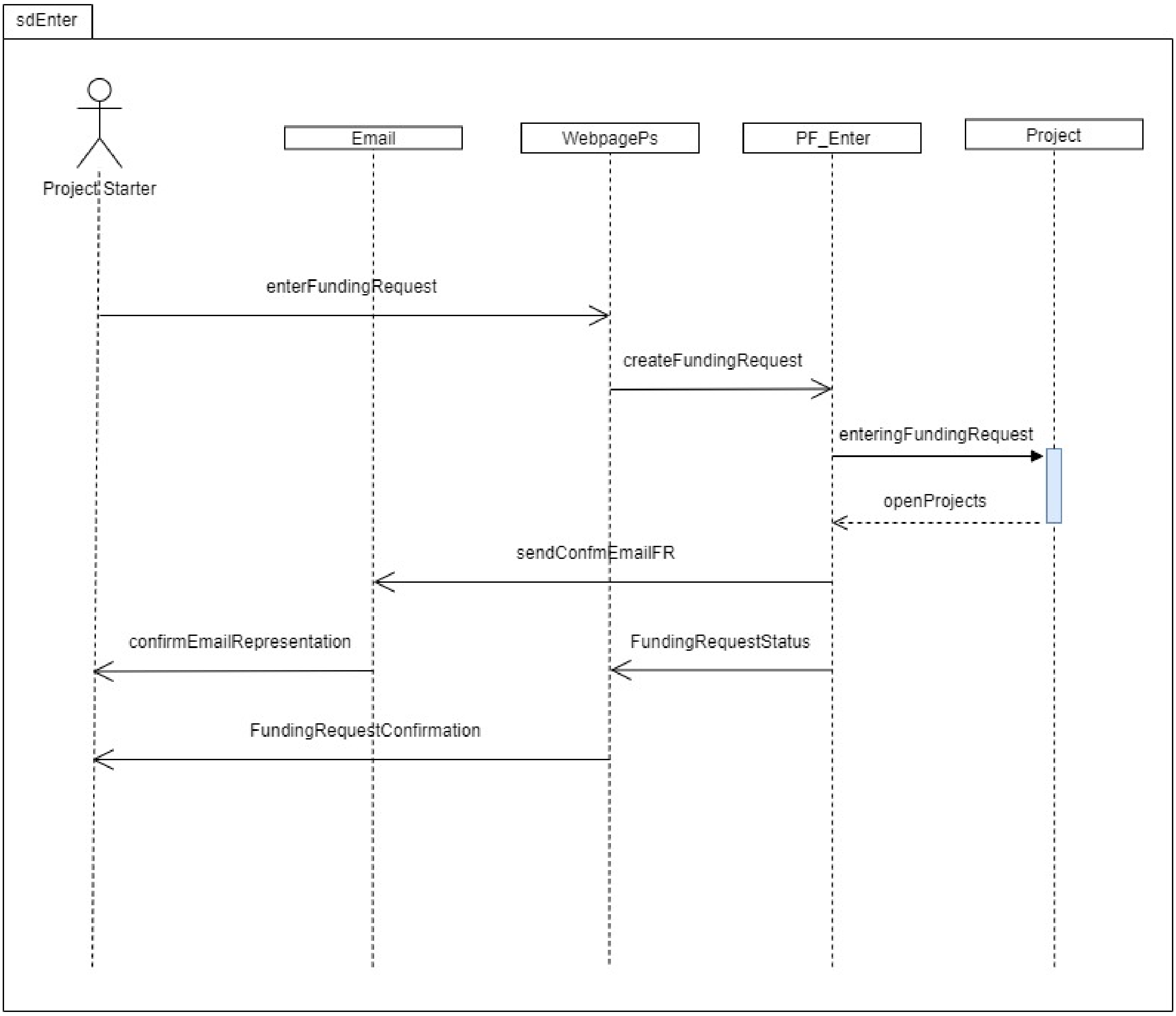


Figure 1.3.1: Sequence Diagram for (S01)

Diagram Description:

Here Project Startersend command “enterFundingRequest” to WebpagePS and then it will command “createFundingRequest” to Machine and Machine will forward the command “enteringFundingRequest” to project. Then the open Projects will be shown and this

confirmation of entering funding request will be sent by Email to Project Starter.

Sequence Diagram for specification (S04):

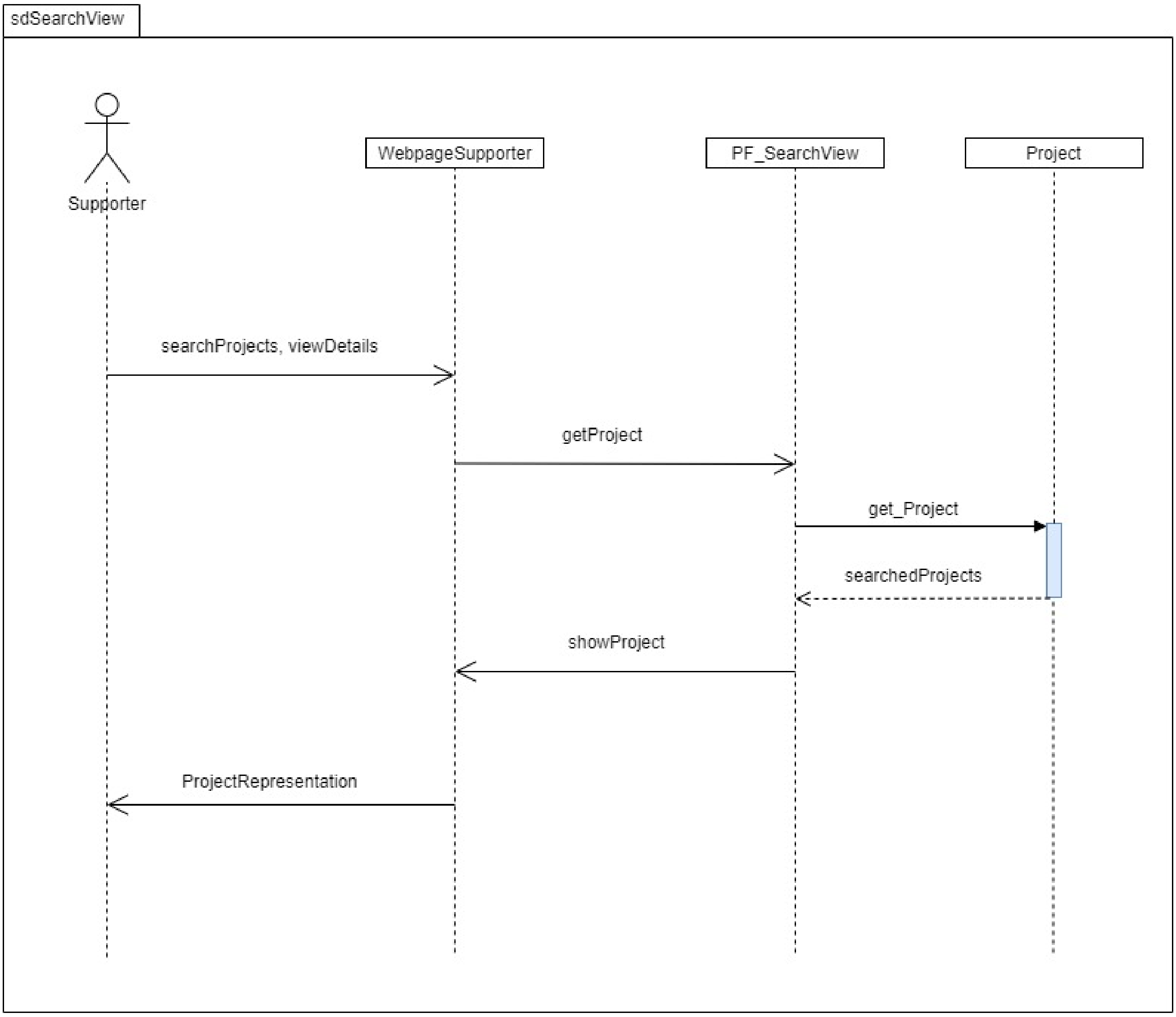


Figure 1.3.2: Sequence Diagram for (S04)

Diagram Description:

Supporters sends the command "searchProject, viewDetails" to WebpageSupporter. After receiving the command the connectionDomain will forward a command "getProject" to Machine. The Machine will send message "get_Project" to Project and Project will responds with "searchedProjects". Then machine will send the command "showProject" to connectionDomain and the Supporter will be able to get the command" ProjectRepresentation by WebpageSupporter.

Sequence Diagram for specification (S07):

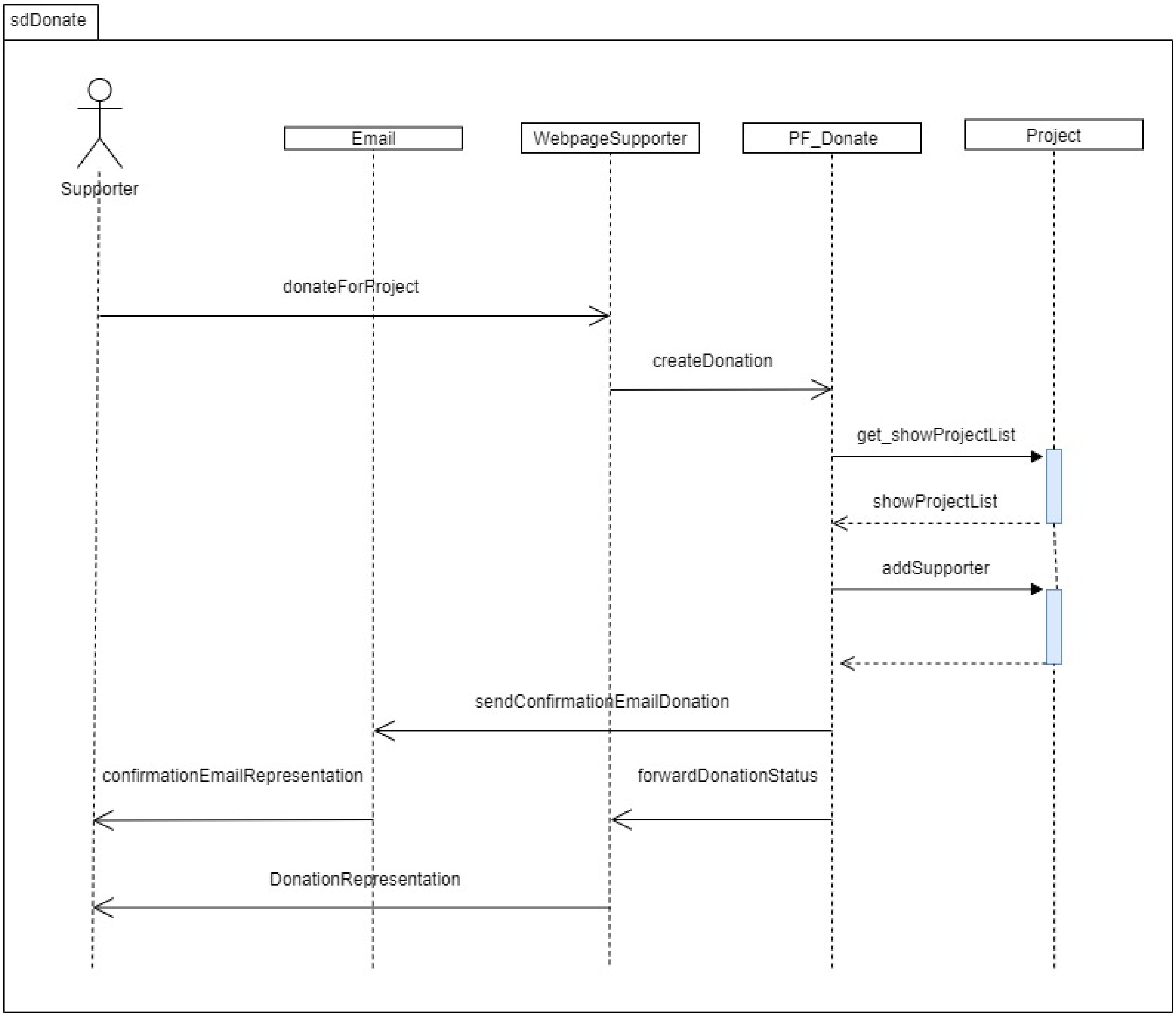


Figure 1.3.2: Sequence Diagram for (S07)

Diagram Description:

Here the diagram is for donating for a project. At first Supporter will send he command "donateForProject" to connectionDomain. Then WebpageSupporter will forward the command "createDonation" to Machine and Machine will send a message " get_showProjectList" to Project. After getting the message, Project will reply to Machine with the command "showProjectList". Then machine will send command "addProject" for adding the liking projects of Supporters. Then Confirmation will be sent via displayDomain Email to Supporters.

Sequence Diagram for specification (S10):

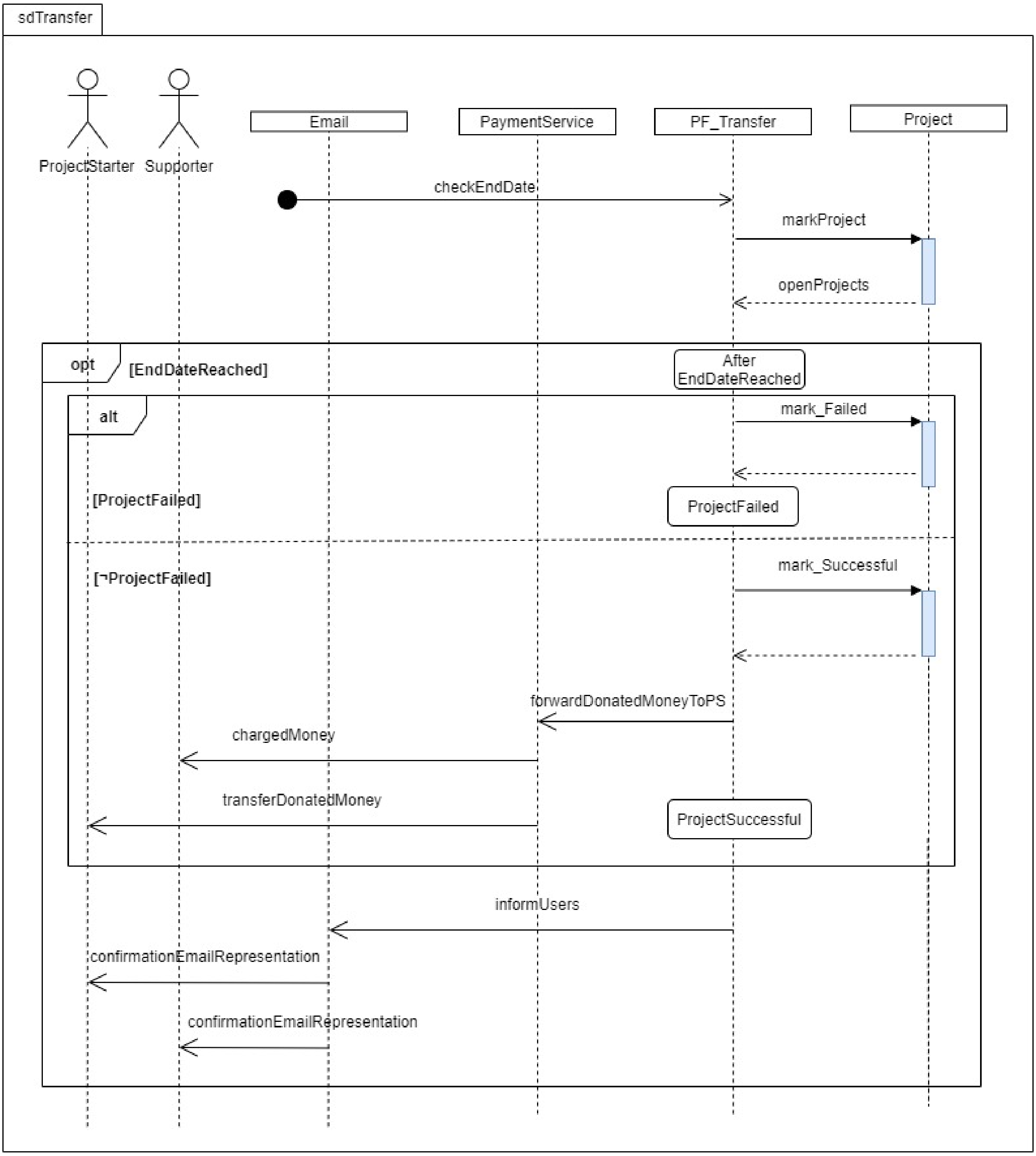


Figure 1.3.2: Sequence Diagram for (S10)

Diagram Description:

After getting a message “checkEndDate”, the Machine will send the command “markProject” to project and if the funding limit was reached the Machine will mark the project Successful and the PaymentService will charged the money from Supporter and transfer the donated money to Project Starter after receiving the command from Machine. In both cases of success and failure, the machine will inform the users by using the displayDomain Email.

1.3.3 Validation

- Sabstract ∧ D are non-contradictory. No contradictions can be found in Sabstract ∧ D.
- Sabstract ∧ D =⇒ R.

(S01a) ^ (S01b) ^ (S01c) ^ (F05) =) (R01)

(S04a) ∧ (S04b) ∧ (S04c) =⇒ (R04)

(F06) ^ (S04a) ^ (S04b) ^ (S04c) =) (R04)

(S10a) ^ (S10b) ^ (S10c) ^ (F07) =) (R10)

- Messages and phenomena are consistent.

message in scenario	source	target	phenomena in problem diagram
enterFundingRequest	ProjectStarter	WebpagePS	PS!{enterFundingRequest}
createFundingRequest	WebpagePS	PF_Enter	WPS!{createFundingRequest}
enteringFundingRequest	PF_Enter	Project	PFE!{enteringFundingRequest}
sendConfirmEmailFR	PF_Enter	Email	PFE!{ sendConfirmEmailFR}
confirmEmailRepresentation	Email	ProjectStarter	E!{confirmEmailRepresentation}
FundingRequestStatus	PF_Enter	WebpagePS	PFE!{FundingRequestStatus}
FundingRequestConfirmation	WebpagePS	ProjectStarter	WPS!{FundingRequestConfirmation}
searchProjects, viewDetails	Supporter	WebpageSupporter	S!{ searchProjects, viewDetails}
getProject	WebpageSupporter	PF_SearchView	WS!{getProject}
get_Project	PF_SearchView	Project	P!{searchedProjects}
showProject	PF_SearchView	WebpageSupporter	PFSV!{showProject}
ProjectRepresentation	WebpageSupporter	Supporter	WS!{ProjectRepresentation}
donateForProject	Supporter	WebpageSupporter	S!{donateForProject}
createDonation	WebpageSupporter	PF_Donate	WS!{createDonation}
get_showProjectList	PF_Donate	Project	P!{showProjectList}
addSupporter	PF_Donate	Project	PFD!{addSupporter}
sendConfirmationEmailDonation	PF_Donate	Email	PFD!{sendConfirmationEmailDonation}
confirmationEmailRepresentation	Email	Supporter	E!{confirmationEmailRepresentation}
forwardDonationStatus	PF_Donate	WebpageSupporter	PFD!{forwardDonationStatus}
DonationRepresentation	WebpageSupporter	Supporter	WS!{DonationRepresentation}
checkProject	-	PF_Transfer	timed event
markProject	PF_Transfer	Project	PFT!{markProject}
mark_failed	PF_Transfer	Project	P!{showProjectList}
mark_Successful	PF_Transfer	Project	P!{showProjectList}
forwardDonatedMoneyToPS	PF_Transfer	PaymentService	PFT!{forwardDonatedMoneyToPS}
chargedMoney	PaymentService	Supporter	PS!{chargedMoney}
transferDonatedMoney	PaymentService	ProjectStarter	PS!{transferDonatedMoney}
informUsers	PF_Transfer	Email	PFT!{informUsers}
confirmationEmailRepresentation	Email	ProjectStarter	E!{confirmationEmailRepresentation}
confirmationEmailRepresentation	Email	Supporter	E!{confirmationEmailRepresentation}

- Lexical domains are not sources of messages.

message in scenario	source	domain type
enterFundingRequest createFundingRequest enteringFundingRequest sendConfirmEmailFR confirmEmailRepresentation FundingRequestStatus FundingRequestConfirmation	ProjectStarter WebpagePS PF_Enter PF_Enter Email PF_Enter WebpagePS	BiddableDomain ConnectionDomain Machine Machine DisplayDomain Machine ConnectionDomain
searchProjects, viewDetails getProject get_Project showProject ProjectRepresentation	Supporter WebpageSupporter PF_SearchView PF_SearchView WebpageSupporter	BiddableDomain ConnectionDomain Machine Machine ConnectionDomain
donateForProject createDonation get_showProjectList addSupporter sendConfirmationEmailDonation confirmationEmailRepresentation forwardDonationStatus DonationRepresentation	Supporter WebpageSupporter PF_Donate PF_Donate PF_Donate Email PF_Donate WebpageSupporter	BiddableDomain ConnectionDomain Machine Machine Machine DisplayDomain Machine ConnectionDomain
checkProject markProject mark_failed mark_Successful forwardDonatedMoneyToPS chargedMoney transferDonatedMoney informUsers confirmationEmailRepresentation confirmationEmailRepresentation	- PF_Transfer PF_Transfer PF_Transfer PF_Transfer PaymentService PaymentService PF_Transfer Email Email	- Machine Machine Machine Machine CausalDomain CausalDomain Machine DisplayDomain DisplayDomain

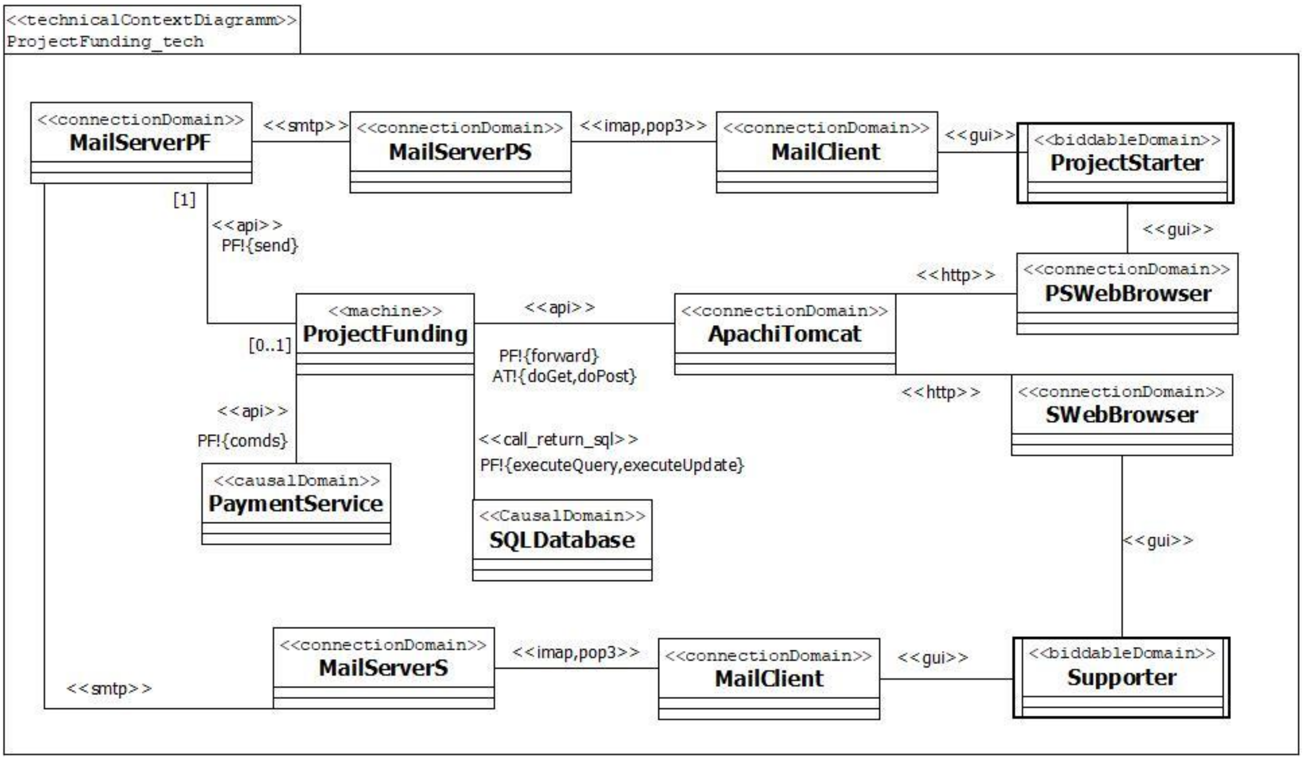
- There exists at least one scenario for each subproblem.
- For each subproblem scenarios exist that consider normal and exceptional cases.

subproblem	normal case	exceptional case
pdEnter	sdEnter	
pdSearchView	sdSearchView	
pdDonate	sdDonate	
pdTransfer	sdTransfer	sdTransfer

Only for pdTransfer an exceptional case must be considered.

1.4 A4

1.4.1 Technical Context Diagram.



<<technicalContextDiagramm>>
ProjectFunding_tech
<<connectionDomain>> MailServerPF
<<smtp>>
<<connectionDomain>> MailServerPS
<<imap,pop3>>
<<connectionDomain>> MailClient
<<gui>>
<<biddableDomain>> ProjectStarter
[1]
<<api>>
PF!{send}
<<gui>>
<<http>>
<<connectionDomain>> PSWebBrowser
<<machine>> ProjectFunding
<<api>>
<<connectionDomain>> ApachiTomcat
[0..1]
PF!{forward}
AT!{doGet,doPost}
<<http>>
<<connectionDomain>> SWebBrowser
<<api>>
PF!{comds}
<<call_return_sql>>
PF!{executeQuery,executeUpdate}
<<causalDomain>> PaymentService
<<CausalDomain>> SQLDatabase
<<gui>>
<<connectionDomain>> MailServerS
<<imap,pop3>>
<<connectionDomain>> MailClient
<<gui>>
<<biddableDomain>> Supporter
<<smtp>>

Figure 1.4.1: Technical Context Diagram

Description:

This is the technical context Diagram of our Project Funding. Here Machine is given command "forward" to ApachiTomcat and get the response as "doGet,doPost". Same machine is given command to MailServerPf "send" vand the command is forwarded via several connectionDomain to Users. Users are using their own browsers. In SQLDatabase "executeQuery, executeUpdate" is commanded by Machine.

1.4.2 Mapping.

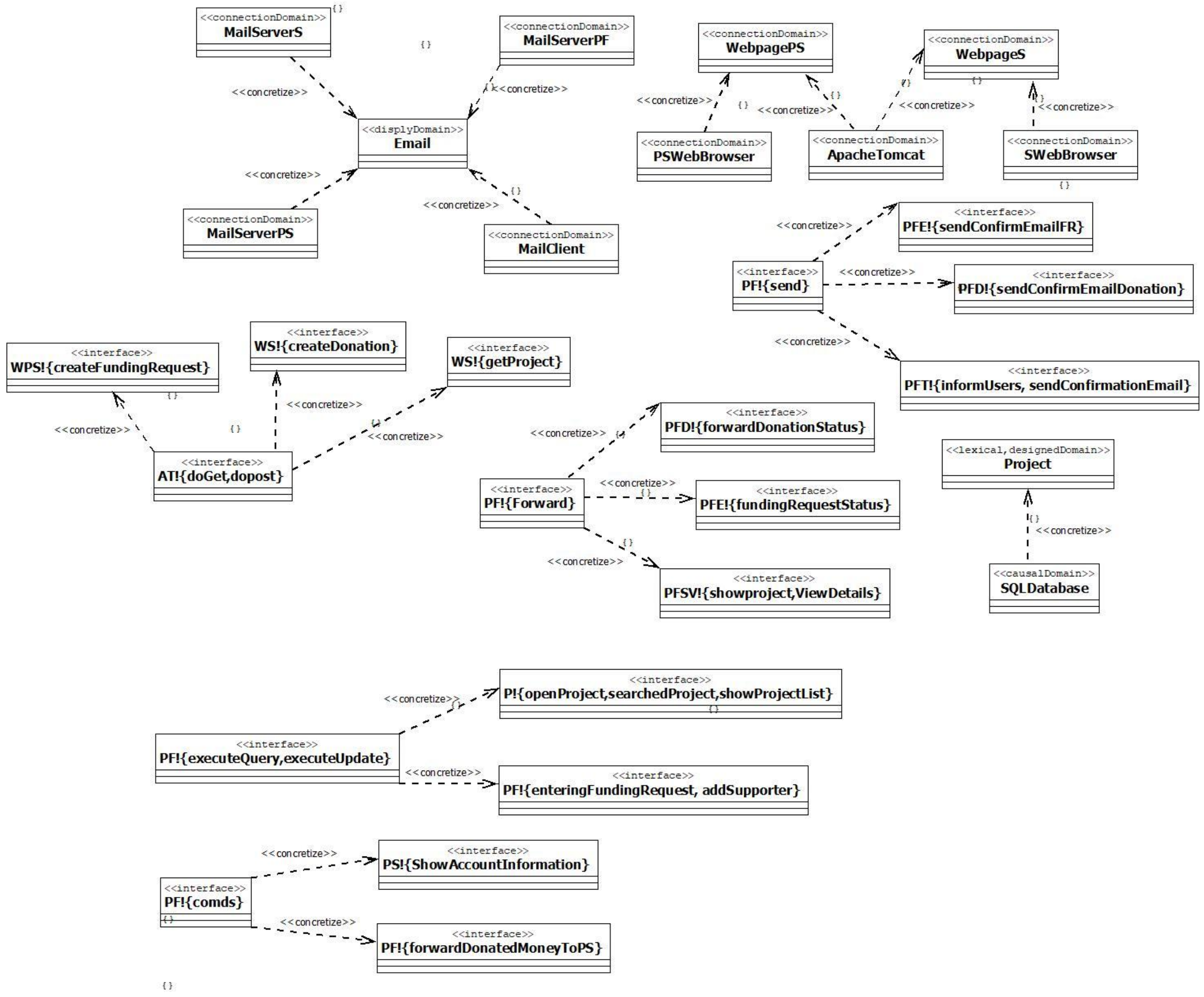


Figure 1.4.1: Mapping for the Technical Context Diagram

Description:

The mapping for technical context diagram where Email display domain concratizes with its MailServer of ProjectFunding, ProjectStarter and Supporter and MailClient. WebpagePS concretizes with its WebBrowser and ApacheTomacat. WebpageS is concretzed with its

WebBrowser and also ApacheTomcat. ApacheTomcat cocretizes with WPS, WS. Project is in SQLDatabase. PF is concratized with the problem Diagram’s sub machine PFD, PFE, PFSV.

1.4.3 Software Specification.

Technical interfaces of the machine:

- API for MailServerVR: Apache Commons Email API (http://commons.apache.org/proper/commons-email/ javadocs/api-release/index.html) Operation send defined in abstract class org.apache.commons.mail.Email
- SQL Commands: defined in FIPS PUB 127-2, (U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 1993) Operations executeQuery and executeUpdate are defined in interface java.sql.Statement (https://docs.oracle.com/javase/8/docs/api/ index.html?java/sql/Statement.html)
- API for ApacheTomcat (http: //tomcat.apache.org/tomcat-9.0-doc/index.html)

  Operations doGet and doPost are defined in abstract class javax.servlet.http.HttpServlet (https://docs.oracle.com/javaee/7/api/javax/ servlet/http/HttpServlet.html)

  Operation forward defined in interface javax.servlet.RequestDispatcher (http://docs.oracle.com/javaee/7/api/javax/ servlet/RequestDispatcher.html)

Technical interfaces in the environment:

- SMTP (Simple Mail Transfer Protocol): defined in Request for Comments (RFC) 2821, (Network Working Group, 2001)
- HTTP (Hypertext Transfer Protocol): defined in RFC 2616, (Network Working Group, 1999)
- IMAP (Internet Message Access Protocol): defined in RFC 3501, (Network Working Group, 2003)
- GUI: User interfaces of MailClient and HTML webpages (defined by https://www.w3.org/TR/html5/) presented by GuestWebBrowser and SMWebBrowser.

1.4.4 Validation.

New phenomena and domains are suitable to implement the
external messages used in the abstract phenomena:

Message	**new phenomena and domains**
createFundingRequest	ApacheTomcat, HTTP
createDonation	ApacheTomcat, HTTP
sendConfirmationEmailFR	SMTP
sendConfirmEmailDonation	SMTP

All internal messages can be realized using SQL commands

- All domains of the technical context diagram are related to domains in the problem diagrams:
- All phenomena in the technical context diagram are related to elements in the problem diagrams:

Provided mapping diagram

All domains directly connected with the machine in the problem diagrams are related to elements in the technical context diagram:

Problem Diagram	**Domain connected with the machine**	**Element in the TCD**
pdEnter	Project	SQLDatabase
	WebpagePS	PSWebBrowser, ApacheTomcat
	Email	MailServerPF, MailServerPS, MailClient
pdSearchView	Project	SQLDatabase
	WebpageSupporter	SWebBrowser, ApacheTomcat
pdDonate	Project	SQLDatabase
	WebpageSupporter	SWebBrowser, ApacheTomcat

	Email	MailServerPF, MailServerS, MailClient
pdTransfer	Project	SQLDatabase
	Email	MailServerPF, MailServerS, MailServerPS, MailClient
	PaymentService	-

1.5 A5

1.5.1 The operation enterFundingRequest (Class model)

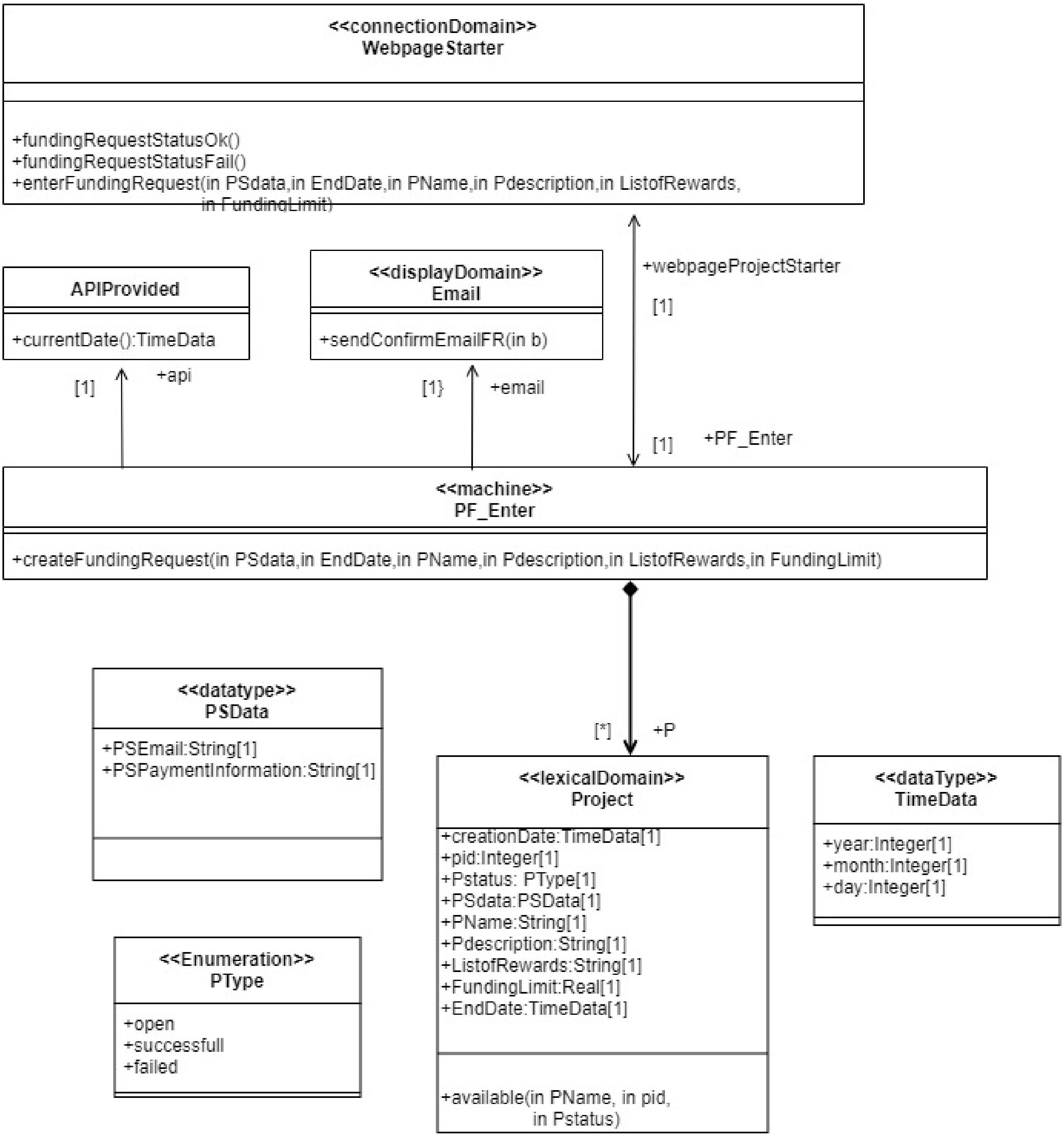


Figure 1.5.1 Class model of operation enterFundingRequest.

Name: enterFundingRequest

Description: Forwards the enter funding request from the Project Starter to the machine.

OCL constraint:

```
context WebpagePS :: enterFundingRequest(PSdata : PSData, EndDate : TimeData,
PName:String, Pdescription : String, ListofRewards:String, FundingLimit:Real)
pre :  true
post:  PF_Enter ^ createFundingRequest (PSdata, EndDate, PName, Pdescription,
ListofRewards, FundingLimit)
```

Name: createFundingRequest

Description: Forwards the request „createFundingRequest" from the Project Starter to the machine and returns a confirmation Email with a link.

OCL constraint:

```
context PF_Enter:: createFundingRequest(PSdata : PSData, EndDate : TimeData,
PName:String, Pdescription : String, ListofRewards:String, FundingLimit:Real)
pre :  true
post:
    if
    project->exists(pr:Project | pr.PName = PName and Pr.Pdescription = Pdescription)
    then
    WebpagePS^fundingRequestStatusFail()
    else
    let
    p: Project = p->any(pr:Project | pr.id = pid) in
         p.enterFR->one(enter : enterFR|
         enter. PSdata = PSdata and
         enter. PEndDate = PEndDate and
         enter.PName = PName and
         enter.Pdescription = Pdescription and
         enter.ListofRewards = ListofRewards and
         enter.FundingLimit = FundingLimit and
         email^sendEmailWithLink () and
         WebpagePS^fundingRequestStatusOk() )
     endif
```

Remarks:

- The class enterFR is introduced to represent of creating Project or entering Fundung Request
- The function currentDate() : TimeData is provided externally. Hence, it is not specified further.

As Project has Unique id

OCL constraint:

```
context Project
inv: Project.allInstances() -> isUnique(id)
```

1.5.2 The operation searchProject (Class model)

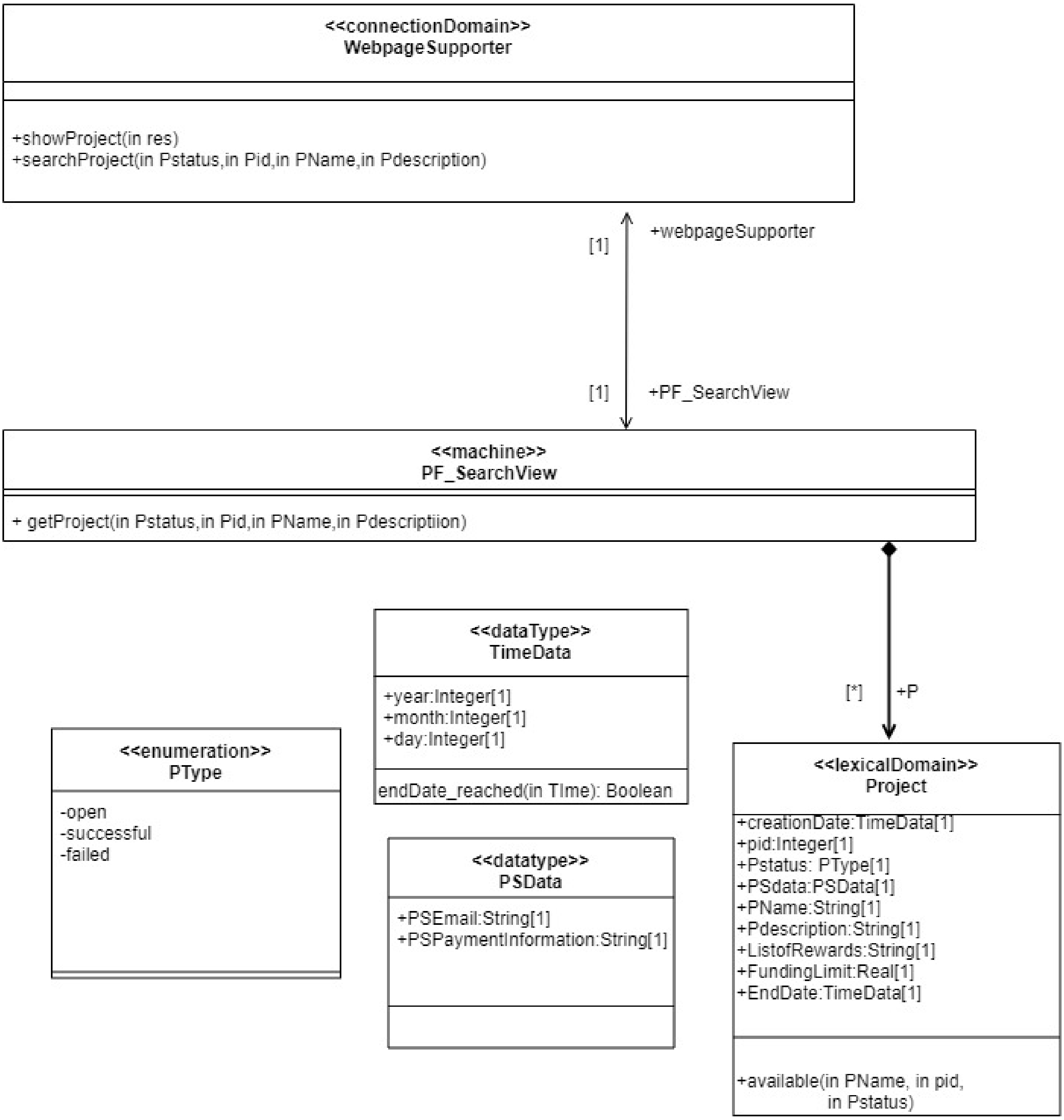


Figure 1.5.2 Class model of operation searchProject.

Name: searchProject

Description: Forwards the search request for open projects from the Supporter to the machine.

OCL constraint:

```
context WebpageSupporter :: searchProject(Pstatus : PType, Pid : Integer, PName :
          String, Pdescription : String)
pre :  true
post:  PF_SearchView ^ getProject(Pstatus, Pid, PName, Pdescription)
```

Name: getProject

Description: Generates and returns a list of Projects matching the input criteria concerning Project Status (open Projects), Project ID, Project Name, Project Description.

OCL constraint:

```
context PF_SearchView :: getProject(Pstatus : PType, Pid : Integer, PName : String,
          Pdescription : String)
pre :  true
post:  let res : Set(Project) = P-> select(p : Project|
        p.Pstatus = Pstatus and
        p.PName = PName and
        p.Pdescription = Pdescription and
        p.Available(PName, Pdescription, Pstatus)) -> asSet()
in
PF_SearchView ^ showProject(res)
```

We want to be able to identify Projects by a unique id.

OCL constraint:

```
context Project
inv: Project.allInstances() -> isUnique(id)
```

1.5.3 The operation donateForProject (Class model)

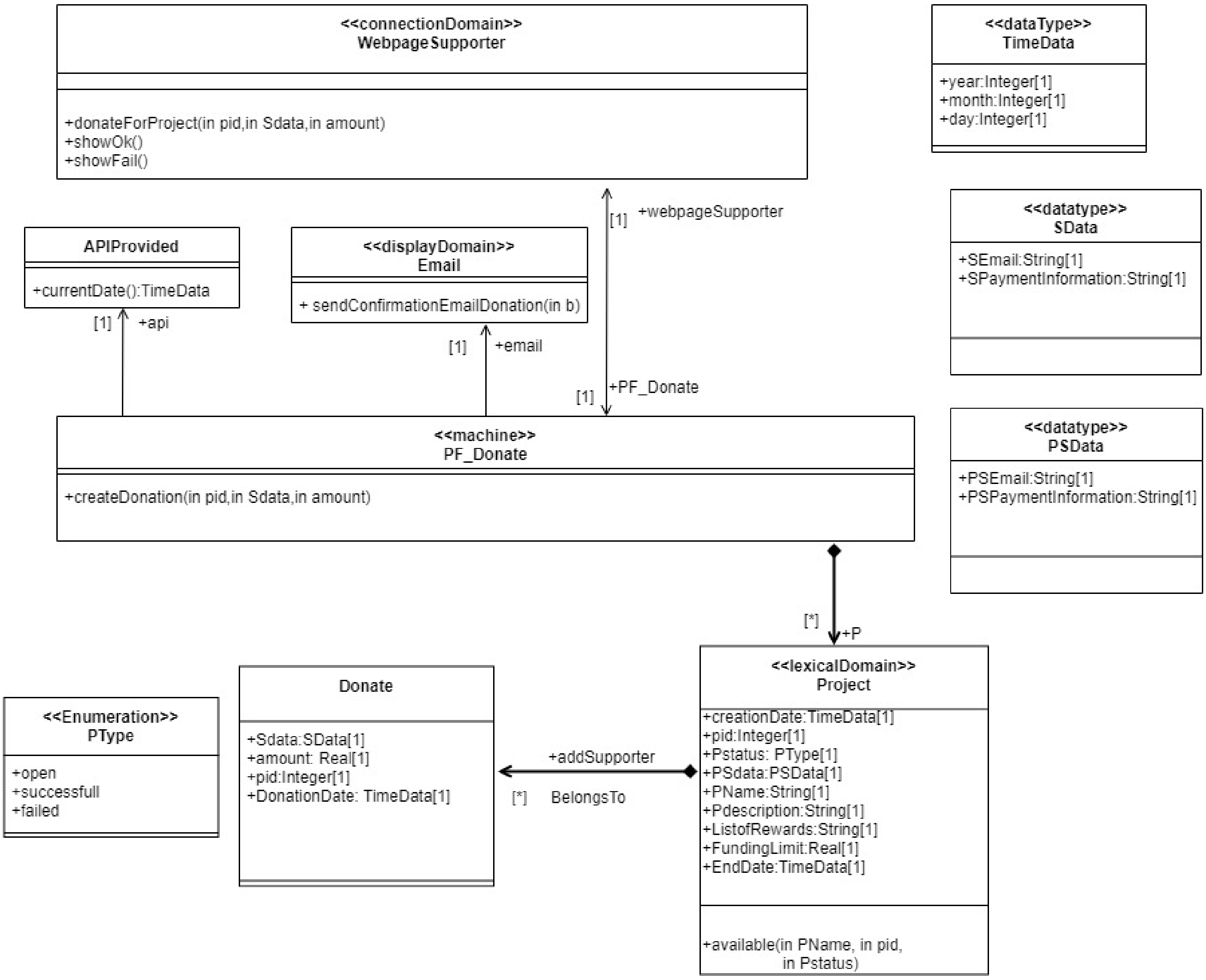


Figure 1.5.3 Class model of operation donateForProject.

Name: donateForProject

Description: Forwards the donate for projects requests from the Supporter to the machine.

OCL constraint:

context WebpageSupporter::donateForProject(pid: Integer, Sdata: SData, amount: Real)
pre : true

```
post:  PF_Donate ^ createDonation(pid, Sdata, amount)
```

Name: createDonation

Description: Donate for project and returns confirmation Email Ok or fails.

OCL constraint:

```
context PF_Donate::createDonation(pid:Integer, Sdata:SData, amount:Real)
pre :  p->one(p:Project|p.id = pid)
post:  let
       p:Project = p->any(pr:Project|pr.id = pid) in
  if p@pre.available(PName, pid, Pstatus)
  then
  p.addSupporter->one(don:Donate|
  don.Sdata = Sdata and
  don.amount = amount and
  don.pid = pid and
  don.DonationDate = api.currentDate() and
  email^sendConfirmationEmailDonation(p.addSupporter->any(don:Donate|
  don.Sdata = Sdata and
  don.amount = amount and
  don.pid = pid)) and
  webpageSupporter^showOk()
  else
  webpageSupporter^showFail()
  endif
```

We want to be able to identify Projects by a unique id.

OCL constraint:

```
context Project
inv: Project.allInstances() -> isUnique(id)
```

As Projects, Donates have a unique id.

OCL constraint:

```
context Donate
inv: Donate.allInstances() -> isUnique(id)
```

1.5.4 The operation checkEndDate (Class model)

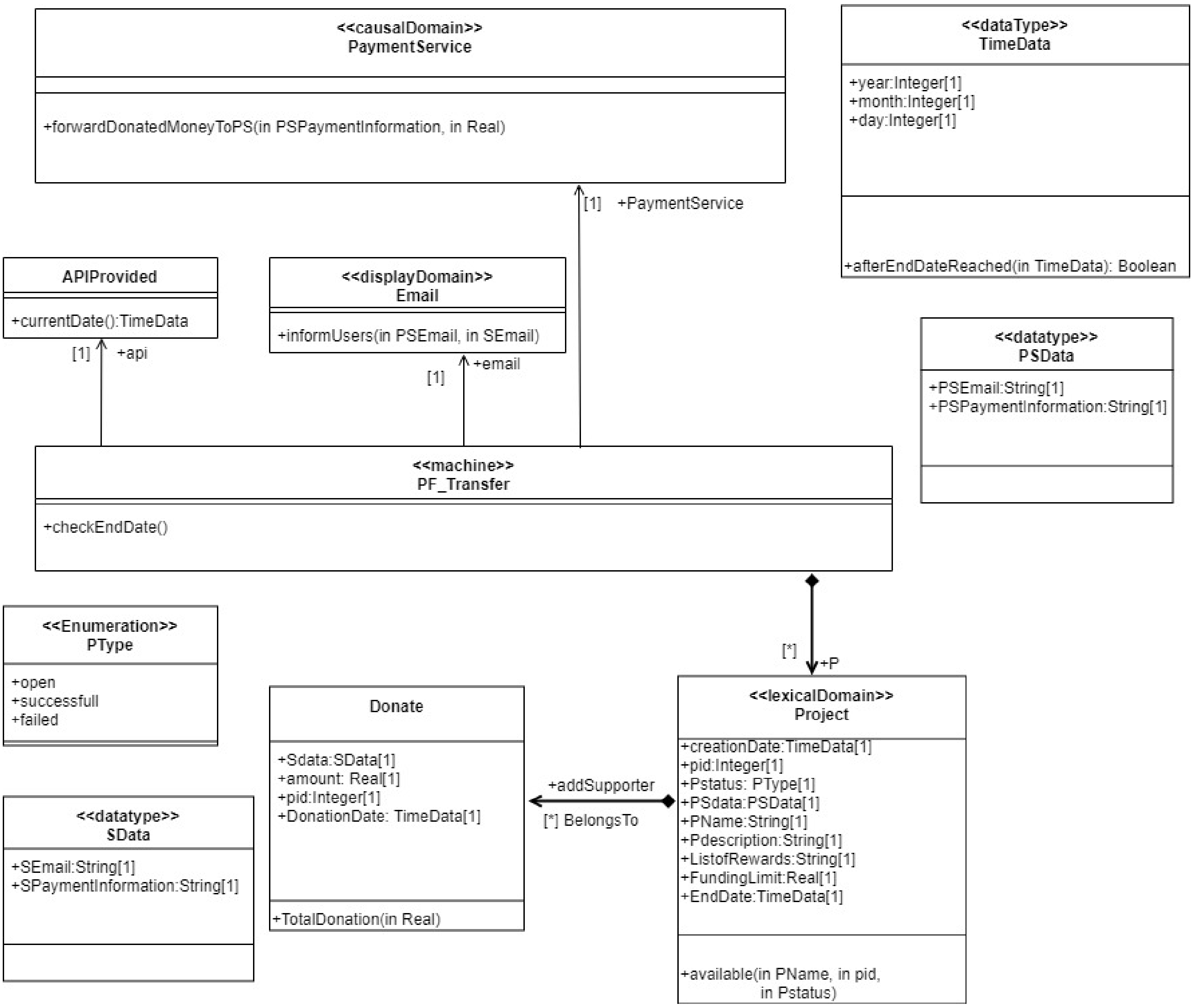


Figure 1.5.4 Class model of operation checkEndDate.

Name: checkEndDate

Description: This internal operation check the end date of a project and if the funding limit was reached then it marks the project. Moreover it transfer the donated money to Project Starter when project is marked as successful.

OCL constraint:

```
context PF_Transfer::checkEndDate()
pre :  true
post:  let
        p:Project = p->any(pr:Project|pr.id = pid) in
   p@pre.afterEndDateReached(EndDate)->asSet() in
   if
   p.addSupporter -> select(don:Donate|
    amount = don.TotalDonation(self.amount))->size() >=self.FundingLimit
   then
   p.Pstatus = PType :: successful and
   email^informUsers(PSData.PSEmail, SData.SEmail) and
PaymentService^forwardDonatedMoneyToPS(PSData.PSPaymentInformation, amount)
   else
   p.Pstatus = PType :: closed and
   email^informUsers(PSData.PSEmail, SData.SEmail)
   endif
```

1.5.5 Validation

Validation for each Subproblem is given here separately.

For enterFundingRequest:

- Operation specifications must be consistent with abstract specification:
  The operation specification of enterFundingRequest is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal and exceptional case behaviour described in the abstract specification are covered in the postcondition.
- Parameters must be used in the pre- and / or postcondition:
  The parameters are used in the pre and postcondition.
- All parameters of operations must be known by the caller and all parameters of sent message must be known by the machine:
  ProjectStarter can input all parameters to WebpagePS via his/her web browser, which forwards these to this operation. The machine knows the enterFR object argument used in the message email.

- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  New class enterFR is added to help hold data while entering a Funding Request and it belongs to Project. Time Date simply is a representation of a date.

For searchProject:

- Operation specifications must be consistent with abstract specification:
  The operation specification of searchProject is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal and exceptional case behaviour described in the abstract specification are covered in the postcondition.
- Parameters must be used in the pre- and / or postcondition:
  The parameters are used in the pre and postcondition.
- All parameters of operations must be known by the caller and all parameters of sent message must be known by the machine:
  Supporter can input all parameters to WebpageSupporter via his/her web browser, which forwards these to this operation.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  An enumeration type class is introduced here for possible project status. Time Date simply is a representation of a date.

For donateForProject:

- Operation specifications must be consistent with abstract specification:
  The operation specification of donateForProject is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal and exceptional case behaviour described in the abstract specification are covered in the postcondition.
- Parameters must be used in the pre- and / or postcondition:
  The parameters are used in the pre and postcondition.

- All parameters of operations must be known by the caller and all parameters of sent message must be known by the machine:
  Supporter can input all parameters to WebpageSupporter via his/her web browser, which forwards these to this operation. The machine knows the addSupporter object argument used in the message email.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  New class Donate is added to help hold data while entering donation and it belongs to Project. Time Date simply is a representation of a date.

For checkEndDate:

- Operation specifications must be consistent with abstract specification:
  The operation specification of checkEndDate is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal and exceptional case behaviour described in the abstract specification are covered in the postcondition.
- Parameters must be used in the pre- and / or postcondition:
  The parameters are used in the pre and postcondition.
- All parameters of operations must be known by the caller and all parameters of sent message must be known by the machine:
  The Operation has no parameters.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  An enumeration type class is introduced here for possible project status. Time Date simply is a representation of a date. afterEndDateReached(time: TimeData): Boolean is added to the data type TimeData because we need to compare modified dates.

1.6 A6

1.6.1 Project Funding life-cycle

$LC_{project\ starter}$ = $(Enter)^{+}$

$LC_{supporter}$ = $((SearchView)^{+}; [Donate])^{*}$

$LC_{project\ funding}$= $(||_{i=1}^{n} LC_{project\ starter})||(||_{j=1}^{m} LC_{supporter})||$Transfer*

Where $||_{i=1}^{n} LC_{i}$ denotes the parallel composition of n copies of life-cycle LC.

1.6.2 Validation

- Each sequence diagram of Step A3: Abstract software specification is contained in at least one life-cycle expression

scenario	life-cycle expression
sdEnter	$LC_{project\ starter}$
sdSearchView	$LC_{supporter}$
sdDonate	$LC_{supporter}$
sdTransfer	$LC_{project\ funding}$

- For all the biddable domains (ProjectStarter and Supporter) exactly one life-cycle exists.
- The life-cycles are consistent with the state predicates in Step A3: Abstract software specification:
  1. Enter has no state predicates at the beginning and end. Hence, it can be executed an arbitrary number of times.
  2. SearchView has no state predicates at the beginning and end. Hence, it can be executed an arbitrary number of times.
  3. Donate can be executed if a project object is created beforehand. Otherwise, SearchView returns an empty set and no project can be selected.
  4. Transfer has no state predicates at the beginning. Hence, it can be executed an arbitrary number of times.

- the life-cycles are consistent with the pre- and postconditions in Step A5: Operations and data specification:
    1. The sequence diagram Enter contains the operation createFundingRequest. It has no precondition. Hence, it can be executed at any position of the life-cycle.
    2. The sequence diagram SearchView contains the operation getProject. It has no precondition. Hence, it can be executed at any position of the life-cycle.
    3. The sequence diagram Donate contains the operation createDonation. createDonation requires, that an project with the supplied pid exists. This is ensured by the postcondition of getProject, that returns a subset of all existing Projects. Only the pid being an element of this list can be used as an input for createDonation. Hence, SearchView must be executed before Donation.
    4. The sequence diagram Transfer contains the operation checkEndDate. It has no precondition. Hence, it can be executed at any position of the life-cycle.

- Exactly one life-cycle exists for the machine domain, that combines all life-cycles The life-cycle $LC_{project\ funding}$ exists for the machine domain. It combines all life-cycles.

Glossary

Name	Type	Description	Source
A			
access to the internet	Phenomenon	When we connect to a platform though a particular network	RD
ApacheTomcat	connectionDomain	An Open Source JSP and Servlet Container from the Apache Foundation.	TCD
API	technical Phenomenon	for MailServerVR: Apache Commons Email API	TCD
APIProvided	class	provides the current date	class model
available()	auxiliary function	it helps to find out whether the project is available or not	class model
B			
be ready for	Phenomenon	To be prepared for something	RD
C			
can cancel open projects or donation	Phenomenon	Project starter can cancel their uploaded project and supporters can cancel their donated money request	CD
can search for	Phenomenon	Where users can search or look for anything	CD, pdSearchView

Name	Type	Description	Source
Confirmation link	Domain	The link which will be sent to the user email in order to confirm the request	RD
currentDate()	auxiliary function	Provides the current Date	class model
checkEndDate()	auxiliary function	it helps to check the end date of a project	class model
create Donation	phenomenon	create the space of Donation	pdDonate
chargedMoney	phenomeon	money has been charged	CD, pdTransfer
createFundingRequest()	auxiliary function	it helps to create a project by storing the infoemation of funding request	class model
createDonation()	auxiliary function	helps to create a donation function	class model
D			
Deadline	Domain	The end time when a contract or task will be finished	RD
Donation	Domain	The amount of money which will be given by Supporters for a project	CD
DonationRepresentation	Domain	represent the Donation	pdDonate
donateForProject()	auxiliary function	in order to donate for the projects	class model
E			

Name	Type	Description	Source
enter a funding request	Phenomenon	User can request for funding of his project	CD, pdEnter
enterFR	class	Represents a funding request	class model
Email	Domain	It is display domain and it is used to send confirmation to users	pdEnter, pdDonate, pdTransfer
EmailAdapter	Component	adapter for using the EmailServerClient	subArchBook, globalArch
EndDateReached	state predicate	Its is the deadline of the project which is already end	sdTransfer
enterFundingRequest()	auxiliary Function	its used to create a project by project starter	class model
EndDate	attribute	end date of a projectc	class model
F			
Fixed time duration	Domain	Time duration which will be fixed	RD
Funding limit	Domain	The limit of amount of money which will be donated	RD
FundingRequestConfirmation	Domain	Confirmation of Funding Request	RD, pdEnter
forwardDonatedMoneyToPS()	auxiliary function	helps to send the donated money to project starter	class model
FundingRequestStatus	phenomena	Status of a funding Request	pdEnter

Name	Type	Description	Source
FundingRequest	class	holds the information of PS and Project	class model
forwardDonationStatus	phenomenon	forward the status of donation	pdDonate
FundingRequestStatusOk()	Auxiliary Function	shows the status Ok	class model
FundingRequestStatusFail()	Auxiliary Function	shows the status Fail	class model
G			
generate Random links	Phenomenon	Random and not unique link will be created by mychine	RD
getProject	Phenomenon	Get the project Informations	pdSearchView
getEmail	phenomenon	get the email	pdTransfer
get_Project	message	Returns the searched projects that match the request	sdSearchView
get_showProjectList	message	Returns the Project lists that match the request	sdDonate
GUI	technical Phenomenon	User interfaces of MailClient and HTML webpages (defined by https://www.w3.org/TR/html5/) presented by PSWebBrowser and SWebBrowser.	TCD
getProject()	auxiliary function	helps to get the searched projects and view them	class model
H			

Name	Type	Description	Source
HTTP	technical Phenomenon	defined in RFC 2616, (Network Working Group, 1999)	TCD
I			
informUsers()	auxiliary function	inform the users for both successful or failed projects	class model
identify or misuse the link	Phenomenon	When a link can be easily guesed and by this one can do wrong thing	RD
id	attribute	represents the unique id of project	class model
IMAP	technical Phenomenon	defined in RFC 3501, (Network Working Group, 2003)	TCD
inform user	Phenomenon	Infrom the user in case any emmergency	CD
L			
LCsupporter	life-cycle	Life-cycle for one supporter	LC
LCproject starter	life-cycle	Life-cycle for one project starter	LC
LCproject funding	life-cycle	Combined life-cycle (all supporter and the internal operation)	LC
M			
markTheProject	phenomenon	mark a project	CD, pdTransfer

Name	Type	Description	Source
mark_failed	message	mark the projects failed when funding limit is not reached	sdTransfer
mark_successful	message	mark the projects successful when funding limit is reached	sdTransfer
MailServerPF	connectionDomain	Mail server of Machine	TCD
MailServerPS	connectionDomain	Mail server of ProjectStarter	TCD
MailServerS	connectionDomain	Mail server of Supporter	TCD
MailClient	connectionDomain	Mail client for users, which helps to read mails	TCD
P			
Project Starter	Domain	User who will do the project	CD, pdSearchView, pdDonate, pdTransfer, pdEnter
PType	class	the class represents the status of a project	class model
PaymentService	attribute	data on payment service	class model
Platform	Domain	Software or Machine	RD
Payment Information	Domain	The informations of the users by which the payments will be done	CD, pdSearchView, pdDonate, pdTransfer, pdEnter

Name	Type	Description	Source
Project	Domain	The task for that require Donation	CD, pdSearchView, pdDonate, pdTransfer, pdEnter
Project Funding	Domain	The machine we will design	CD
Payment Service	Domain	The service by which payment will be done	CD, pdSearchView, pdDonate, pdTransfer, pdEnter
PaymentServiceAdapter	component	responsible to create and maintain tables for all persistent classes	subArchBrowse subArchBook, subArchReset, globalArch
P_Adapter	component	responsible to create and maintain tables for all persistent classes	subArchBrowse subArchBook, subArchReset, globalArch
PF_Enter	Domain	Submacine of PF and this helps to enter funding request	pdEnter
PF_SearchView	Domain	Submachine of PF and this helps to search open Projects and view deatails	pdSearchView
PF_Donate	Domain	Submachine of PF and this helps to like the projects of supporter and add supporters	pdDonate
PF_Transfer	Domain	Submachine of PF and this helps to mark the project and Donation	pdTransfer
ProjectFailed	state predicate	The project is failed after a period of time	sdTransfer

Name	Type	Description	Source
ProjectSuccessful	state predicate	The project is successful after a period of time	sdTransfer
PSGUI	component	web interface for project supporter	subArchBrowse subArchBook, globalArch
PSdata	attribute	data on project starter	class model
PSWebBrowser	connectionDomain	Web browser used by ProjectStarter, e.g. Chrome.	TCD
PSData	class	strore the attrbutes of project starter	class model
PType	class	enumeration class	class model
SData	class	store the attribute of supporter	class model
showOk()	auxiliary function	show whether the statement is ok	class model
showFail()	auxiliary function	show whether the statement is Failed	class model
sendConfirmationEmail **Donation**	auxiliary function	send the confirmation email when the donation is done	class model
R			
regularly check	Phenomenon	When a person always examine a particular thing	RD
S			

Name	Type	Description	Source
Supporter	Domain	User who will donate for the project	CD, pdSearchView, pdDonate, pdTransfer, pdEnter
SupporterGUI	component	web interface for supporter	subArchBrowse subArchBook, globalArch
SWebBrowser	connectionDomain	Web browser used by Supporter, e.g. Mozilla Firfox.	TCD
SMTP	technical Phenomenon	defined in RFC 2821, (Network Working Group, 2001)	TCD
searchedProjectList	Phenomena	Project List which is searched by Supporter	CD, pdSearchView
SQLDatabase	causalDomain	The database we need where project is included.	TCD
sendConfirmationEmail	phenomena	send the link of confirmation via email to users	pdEnter, pdDonate, pdTransfer
showProjectList	phenomenon	show the list of projects	CD, pdTransfer
showAccountInformation	phenomenon	show the informations of account	CD, pdTransfer
sendConfirmEmailFR()	auxiliary function	its send the funding request confirmation email	class model
showProject()	auxiliary function	it helps to show the open projects	class model

Name	Type	Description	Source
searchProject()	auxiliary function	it helps to search the open projects	class model
T			
TimeData	class	class which store the information of time	class model
Timer	reused component	given component initiating the internal operation “checkPayment”	subArchReset, globalArch
TotalDonation()	auxiliary function	It helps to calculate the total donated amount for each project	class model
U			
Users	Domain	The person who uses the system	RD
V			
Valid email	Domain	Email which is correct and accurate	RD
view Details	Phenomenon	In order to see the details	CD, pdSearchView
W			
Web browser	Domain	A software application for accessing information on the World Wide Web	RD

Name	Type	Description	Source
WebpagePS	Domain, class	Project Supporter in Web Platform	pdEnter, class model
WebpageSupporter	Domain, class	Supporter in Web Platform	pdSearchView, class model
webpageProjectStarter	attribute	data on PS	class model
webpageStarter	attribute	data on Supporter	class model